SYBASE®

Справочное руководство. Том 2. Команды.

Adaptive Server Enterprise

12.5

ИДЕНТИФИКАТОР ДОКУМЕНТА: DC30009-01-1250-01

ПОСЛЕДНЕЕ ОБНОВЛЕНИЕ: Сентябрь 2002

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Содержание

ГЛАВА 7

Команды

В этом томе описываются команды, инструкции и другие элементы, из которых состоят операторы Transact-SQL.

Обзор

В таблице 7-1 приведено краткое описание команд, рассматриваемых в этой главе.

Таблица 7-1. Команды Transact-SQL

Команда	Описание
alter database	Увеличивает размер пространства, выделяемого для базы данных.
alter role	Задает отношения взаимного исключения между ролями, а также добавляет, удаляет и изменяет пароли для ролей.
alter table	Добавляет новые столбцы; добавляет, изменяет или удаляет ограничения; секционирует или десекционирует существующие таблицы.
begin...end	Делает последовательность операторов SQL единым блоком, в результате чего управляющие конструкции (например, if...else) могут работать с этой последовательностью как с единым целым.
begin transaction	Устанавливает начало пользовательской транзакции.
break	Вызывает выход из цикла while. Команда break часто активируется по условию if.
case	Позволяет выбирать выражения SQL для исполнения на основании указанных условий. Выражения case можно применять везде, где могут использоваться выражения, возвращающие значение.
checkpoint	Записывает на устройство хранения базы данных все *грязные* страницы (страницы, которые были изменены с момента последней записи).
close	Деактивирует курсор.
coalesce	Позволяет выбирать выражения SQL для исполнения на основании указанных условий. Выражения coalesce можно применять везде, где могут использоваться выражения, возвращающие значение; альтернатива выражению case.
commit	Устанавливает конец пользовательской транзакции.
compute	Генерирует итоговые значения, которые выводятся как дополнительные строки в результатах запроса.
connect to...disconnect	Указывает сервер, с которым необходимо установить транзитное соединение.

Команда	Описание
continue	Вызывает перезапуск цикла while. Команда continue часто активируется по условию if.
create database	Создает новую базу данных.
create default	Задает значение для вставки в столбец (или во все столбцы определенного пользователем типа данных), если во время вставки в явной форме не было указано никакого значения.
create existing table	Подтверждает, что информация текущей удаленной таблицы соответствует информации, которая хранится в *списке_столбцов*, и проверяет существование базового объекта.
create index	Создает индекс по одному или нескольким столбцам таблицы.
create plan	Создает абстрактный план запроса.
create procedure	Создает хранимую процедуру, которая может принимать один или несколько введенных пользователем параметров.
create proxy_table	Создает прокси-таблицу без указания списка столбцов. Службы Component Integration Services извлекают список столбцов из метаданных, полученных из удаленной таблицы.
create role	Создает пользовательскую роль.
create rule	Задает область допустимых значений для определенного столбца или для любого столбца определенного пользователем типа данных.
create schema	Создает коллекцию таблиц, представлений и полномочий для пользователя базы данных.
create table	Создает новые таблицы, а также может создавать ограничения целостности.
create trigger	Создает триггер – тип хранимой процедуры, часто используемый для реализации ограничений целостности. Триггер автоматически выполняется, когда пользователь пытается выполнить указанный оператор изменения данных в указанной таблице.
create view	Создает представление, которое является одним из способов просмотра данных в одной или нескольких таблицах.
dbcc	Приложение Database Consistency Checker (dbcc) проверяет логическую и физическую целостность базы данных. Команда dbcc должна регулярно использоваться для периодической проверки, а также при подозрениях на какое-либо повреждение базы данных.
deallocate cursor	Делает курсор недоступным и освобождает все ресурсы памяти, выделенные этому курсору.
declare	Объявляет имя и тип локальных переменных для пакета или процедуры.
declare cursor	Определяет курсор.
delete	Удаляет строки из таблицы.
delete statistics	Удаляет статистику из системной таблицы sysstatistics.
disk init	Делает физическое устройство или файл доступным для сервера Adaptive Server.

Команда	Описание
disk mirror	Создает зеркальную копию файлов баз данных, которая сразу включается в работу в случае сбоя первичного устройства.
disk refit	Перестраивает системные таблицы sysusages и sysdatabases базы данных master в соответствии с информацией, содержащейся в таблице sysdevices. Команда disk refit используется после команды disk reinit в комплексе действий по восстановлению базы данных master.
disk reinit	Перестраивает системную таблицу sysdevices базы даных master. Команда disk reinit используется в комплексе действий по восстановлению базы данных master.
disk remirror	Возобновляет зеркальное копирование диска после того, как оно было остановлено из-за сбоя устройства, копия которого создается, или приостановлено командой disk unmirror.
disk unmirror	Отключает исходное устройство или его зеркальную копию для обслуживания или замены оборудования.
drop database	Удаляет одну или несколько баз данных из СУБД Adaptive Server.
drop default	Удаляет определенные пользователем значения по умолчанию.
drop index	Удаляет индекс из таблицы в текущей базе данных.
drop procedure	Удаляет определенные пользователем хранимые процедуры.
drop role	Удаляет определенную пользователем роль.
drop rule	Удаляет определенное пользователем правило.
drop table	Удаляет из базы данных определение таблицы и все относящиеся к ней данные, индексы, триггеры и спецификации полномочий.
drop trigger	Удаляет триггер.
drop view	Удаляет из текущей базы данных одно или несколько представлений.
dump database	Создает резервную копию всей базы данных (включая журнал транзакций) в форме, доступной для чтения командой load database. Создание резервных копий и их загрузка выполняется через сервер Backup Server.
dump transaction	Делает копию журнала транзакций и удаляет его неактивную часть.
execute	Запускает системную процедуру, определенную пользователем хранимую процедуру или динамически созданную команду Transact-SQL.
fetch	Возвращает строку или набор строк из результирующего набора курсора.
goto label	Вызывает переход к определенной пользователем метке.
grant	Предоставляет полномочия пользователям или пользовательским ролям.
group by и having	Используются в операторах select для деления таблицы на группы и для возврата только тех групп, которые удовлетворяют условиям инструкции HAVING.
if...else	Задает условия выполнения оператора SQL.
insert	Вставляет новые строки в таблицу или представление.
kill	Уничтожает процесс.

Команда	Описание
load database	Загружает резервную копию пользовательской базы данных, включая ее журнал транзакций.
load transaction	Загружает резервную копию журнала транзакций.
lock table	Явным образом блокирует таблицу из транзакции.
nullif	Позволяет выбирать выражения SQL для исполнения на основании указанных условий. Выражения nullif можно применять везде, где могут использоваться выражения, возвращающие значение; альтернатива выражению case.
online database	Помечает базу данных как общедоступную после обычной процедуры загрузки и, при необходимости, обновляет загруженные резервные копии базы данных и журнала транзакций до текущей версии сервера Adaptive Server.
open	Открывает курсор для обработки.
order by	Возвращает результаты запроса, отсортированные в указанном порядке по значениям одного или нескольких столбцов.
prepare transaction	Используется приложением DB-Library™, реализующим механизм двухфазной фиксации, для определения готовности сервера к фиксации транзакции.
print	Выводит определенное пользователем сообщение на экран пользователя.
quiesce database	Приостанавливает и возобновляет обновление баз данных из указанного списка.
raiserror	Выводит определенное пользователем сообщение об ошибке на экран пользователя и устанавливает системный флаг для регистрации возникновения ошибки.
readtext	Считывает указанное количество байтов или символов в значениях типа text и image, начиная с заданного смещения.
reconfigure	Команда reconfigure в настоящий момент не работает. Она оставлена лишь для того, чтобы существующие сценарии можно было выполнять без изменения. В предыдущих версиях команда reconfigure вызывалась после системной процедуры sp_configure, чтобы ввести в действие новые значения параметров конфигу-рации.
remove java	Удаляет из базы данных один или несколько классов, пакетов или архивов (JAR) Java-SQL. Используется, если в базе данных включена поддержка языка Java.
reorg	Освобождает неиспользуемое пространство на страницах, устраняет перемещения строк или переписывает все строки таблицы на новые страницы в зависимости от указанного параметра.
return	Выполняет безусловный выход из пакета или процедуры и дополнительно позволяет передать состояние возврата. Операторы, идущие за командой return, не выполняются.
revoke	Отзывает полномочия или роли у пользователей или ролей.
rollback	Выполняет откат пользовательской транзакции к ее последней точке сохранения или к началу транзакции.

Команда	Описание
rollback trigger	Выполняет откат результатов выполнения триггера, включая изменения, вызвавшие запуск триггера, с возможным запуском оператора raiserror.
save transaction	Устанавливает в транзакции точку сохранения.
select	Извлекает строки из объектов базы данных.
set	Устанавливает параметры обработки запросов сервером Adaptive Server на время рабочего сеанса пользователя. Может использоваться для установки параметров в триггере или хранимой процедуре. Кроме того, может применяться для активации или деактивации роли в текущем сеансе.
setuser	Позволяет владельцу базы данных действовать от лица другого пользователя.
shutdown	Прекращает работу сервера Adaptive Server или Backup Server™. Эта команда может выполняться только системным администратором.
truncate table	Удаляет все строки из таблицы.
union	Возвращает один результирующий набор, который объединяет результаты нескольких запросов. Дублирующиеся строки удаляются из результирующего набора, если не указано ключевое слово all.
update	Изменяет данные в существующих строках, вставляя новые данные или изменяя существующие; обновляет всю статистическую информацию о данной таблице; обновляет информацию о числе страниц в каждой секции секционированной таблицы; обновляет информацию о распределении значений ключей в указанных индексах.
use	Указывает базу данных, с которой будет работать пользователь.
waitfor	Указывает конкретное время, временной интервал или событие, при наступлении которых выполняются блок операторов, хранимая процедура или транзакция.
where	Задает условия поиска в операторах select, insert, update и delete.
while	Задает условие повтора выполнения оператора или блока операторов. Операторы продолжают выполняться, пока соблюдается указанное условие.
writetext	Позволяет в диалоговом режиме вносить изменения в существующие столбцы типа данных text и image без отражения этих действий в журнале.

alter database

Описание

Увеличивает пространство, выделенное под базу данных.

Синтаксис

alter database *имя_базы_данных*
[on {default | *устройство_хранения_базы_данных* } [= size]
[, устройство_хранения_базы_данных [= *размер*]]...]
[log on { default | *устройство_хранения_базы_данных* } [= *размер*]
[, *устройство_хранения_базы_данных* [= *размер*]]...]
[with override]
[for load]
[for proxy_update]

Параметры

имя_базы_данных

Имя базы данных. Имя базы данных может быть литералом, переменной или параметром хранимой процедуры.

on

Указывает размер и/или местоположение для расширения базы данных. Если журнал и данные находятся на разных фрагментах устройства, то для устройства хранения данных используется эта инструкция, а для устройства хранения журнала – инструкция log on.

default

Указывает, что команда alter database может разместить расширение базы данных на любом устройстве (устройствах) хранения базы данных, назначенном по умолчанию (см. процедуру sp_helpdevice). Чтобы указать размер расширения базы данных, не указывая точного местоположения, используется следующая команда:

```
on default = размер
```

Чтобы изменить состояние устройства хранения базы данных на состояние по умолчанию, используется системная процедура sp_diskdefault.

устройство_хранения_базы_данных

Имя устройства хранения базы данных, на котором будет размещено расширение базы данных. База данных может занимать несколько устройств хранения, на каждом из которых объем занимаемого пространства может быть различным. Чтобы добавить устройство хранения базы данных к СУБД Adaptive Server, используется команда disk init.

размер

Объем пространства, выделяемого под расширение базы данных. Его можно указать в следующих единицах: 'k' или 'K' (килобайты), 'm' или 'M' (мегабайты), 'g' или 'G' (гигабайты). Sybase рекомендует всегда указывать единицу измерения. Если значение не указано, команда alter database расширяет базу данных на величину, равную максимуму из 1 МБ и 4 единиц выделения пространства. В следующей таблице перечислены минимальные размеры расширения базы данных:

Размер логической страницы сервера	Размер расширения
2 КБ	1 МБ
4 КБ	1 МБ
8 КБ	2 МБ
16 КБ	4 МБ

log on

Указывает, что необходимо выделить дополнительное пространство для журналов транзакций базы данных. Инструкция log on использует те же значения по умолчанию, что и инструкция on.

with override

Заставляет СУБД Adaptive Server принять указанное описание устройства, даже если в результате этого данные и журналы транзакций будут располагаться на одном устройстве, что поставит под угрозу возможность восстановления базы данных на текущий момент времени. Если попытаться разместить журнал и данные на одном устройстве, не указав эту инструкцию, команда alter database не будет выполнена. Если разместить журнал и данные на одном устройстве, указав инструкцию with override, будет выдано предупреждение, но команда будет выполнена.

for load

Используется только после команды create database for load, когда нужно заново выделить пространство и настроить параметры использования сегментов для базы данных, загружаемой из резервной копии.

for proxy_update

Вызывает ресинхронизацию прокси-таблиц в прокси-БД.

Примеры

Пример 1. Добавление 1 МБ пространства к базе данных mydb на устройстве хранения базы данных, заданном по умолчанию:

```
alter database mydb
```

Пример 2. Добавление 3 МБ к пространству, выделенному под базу данных pubs2 на устройстве хранения базы данных newdata:

```
alter database pubs2
on newdata = 3
```

Пример 3. Добавление 10 МБ пространства для данных на устройстве userdata1 и 2 МБ для журнала на устройстве logdev:

```
alter database production
on userdata1 = 10
log on logdev = 2
```

Использование

Ограничения

- Кавычки не обязательны, если единицы измерения не указаны. Но если единицы измерения заданы, то кавычки обязательны.
- Если единицы измерения не указаны, то СУБД Adaptive Server считает, что размер задан в мегабайтах дискового пространства, и пересчитывает это число в логические страницы используемого сервером размера.
- СУБД Adaptive Server выдает ошибку, если суммарный размер всех столбцов фиксированной длины плюс служебная часть строки больше, чем допускают схема блокировки таблицы и размер страницы.
- Если при создании таблицы с блокировкой только данных (data-only locking, DOL) смещение для столбца переменной длины превысит 8191 байт, то в него невозможно будет добавить строки.
- Поскольку при выполнении команд create database и alter database пространство для баз данных выделяется порциями по 256 логических страниц, эти команды округляют указанный размер в меньшую сторону до ближайшего числа, кратного единице выделения памяти.
- Можно задать *размер* как значение типа float, но оно будет округлено в меньшую сторону до ближайшего числа, кратного единице выделения памяти.
- Минимальное пространство, выделяемое под базу данных, равно наибольшему из следующих значений:
  - один мегабайт;
  - одна единица выделения пространства с размером логической страницы сервера.
- В следующих случаях таблицы будут созданы, но при последующих операциях DML над ними будут выданы сообщения об ошибках, связанных с ограничениями на размер:

  - общий размер строк со столбцами переменной длины превышает максимальный размер столбца;
  - длина одного столбца переменной длины превышает максимальный размер столбца;
  - в таблице DOL смещение одного из столбцов переменной длины (кроме начального столбца) превышает 8191 байт.
- Если Adaptive Server не может выделить требуемое пространство, он выделяет максимально возможное на каждом устройстве и выдает сообщение о том, сколько пространства было выделено на каждом устройстве хранения базы данных.
- Команду alter database можно выполнять только из базы данных master (или включить ее в хранимую процедуру, выполняемую из базы данных master).
- Для расширения базы данных master можно использовать только главное устройство (устройство master). Если попытаться использовать команду alter database для расширения базы данных master на любое другое устройство хранения базы данных, будет выведено сообщение об ошибке. Ниже дан пример правильной команды для изменения базы данных master на главном устройстве:

```
alter database master on master = 1
```

- Пространство, выделяемое на устройстве хранения базы данных командами create database или alter database, выделяется фрагментами, а в таблицу sysusages добавляется строка с соответствующей информацией.
- Если команда alter database изменяет базу данных, для которой в данный момент создается резервная копия, команда alter database не может завершиться до того, как закончится резервное копирование. На время резервного копирования блокируется хранящаяся в памяти карта использования пространства базы данных. Если запустить команду alter database в тот момент, когда эта карта заблокирована, эта карта будет обновлена с диска по завершении резервного копирования. Если прервать выполнение команды alter database, то Adaptive Server укажет, что нужно выполнить системную процедуру sp_dbremap. Если не сделать этого, то добавленное пространство будет недоступно серверу Adaptive Server до следующего перезапуска.
- Команду alter database on устройство_хранения_базы_данных можно применять к автономной базе данных.

Резервирование базы данных *master* после выделения дополнительного пространства

- После каждого выполнения команды alter database рекомендуется резервировать базу данных master при помощи команды dump database. Это позволит легко и безопасно восстановить базу данных master в случае ее повреждения.
- Если была использована команда alter database и не удалось создать резервную копию базы данных master, то изменения, скорее всего, можно восстановить командой disk refit.

Размещение журнала на отдельном устройстве

- Увеличить пространство, выделенное для хранения журнала транзакций с помощью расширения log on команды create database, можно, задав имя устройства хранения этого журнала в инструкции log on команды alter database.
- Если для размещения журналов на отдельном устройстве не было использовано расширение log on команды create database, то полное восстановление после сбоя жесткого диска может оказаться невозможным. В таком случае можно расширить журналы, выполнив команду alter database с инструкцией log on, а затем – системную процедуру sp_logdevice.

Получение справки по использованию пространства

- Получить имена, размеры и сведения об использовании фрагментов устройства, используемых базой данных, можно с помощью процедуры sp_helpdb имя_базы_данных.
- Сведения о том, какое пространство использует текущая база данных, можно получить, выполнив процедуру sp_spaceused.

Сегменты *system* и *default*

- Сегменты system и default назначаются каждому новому устройству хранения базы данных, указанному в инструкции on команды alter database. Чтобы отменить назначение этих сегментов, нужно выполнить процедуру sp_dropsegment.
- Если команда alter database (без инструкции override) используется для расширения базы данных на устройство, уже используемое этой базой данных, то сегменты, назначаемые этому устройству, также расширяются. Если инструкция override указана, все фрагменты устройства, указанные в инструкции on, становятся сегментами system/default, а все фрагменты устройства, указанные в инструкции log on, – сегментами журнала.

Использование команды *alter database* для возвращения приостановленных процессов в активное состояние

- Если пользовательские процессы приостанавливаются из-за того, что они достигли последнего порога на сегменте журнала, то можно добавить пространство к этому сегменту журнала при помощи команды alter database. Как только объем свободного пространства превысит последний порог, процессы снова станут активными.

Использование инструкции *for proxy_update*

- Если инструкция for proxy_update введена без других параметров, размер базы данных не будет увеличен; вместо этого прокси-таблицы, если таковые имеются, будут удалены из прокси-БД и воссозданы по метаданным, полученным по пути, указанному во время выполнения команды create database ... with default_location = 'путь'.
- Если эта команда вместе с другими параметрами используется для увеличения размера базы данных, то синхронизация прокси-таблицы выполняется после расширения.
- Цель такого расширения команды alter database – предоставить администратору базы данных удобную одношаговую операцию, с помощью которой можно получить точное и актуальное прокси-представление всех таблиц на некотором удаленном узле.
- Такая ресинхронизация поддерживается для всех внешних источников данных, а не только для первичного сервера в отказоустойчивом кластере. Кроме того, база данных не обязательно должна быть создана с инструкцией for proxy_update. Если местоположение по умолчанию для хранилища данных указано в команде create database или в процедуре sp_defaultloc, то метаданные, содержащиеся в этой базе данных, можно синхронизировать с метаданными в удаленном хранилище.
- Чтобы убедиться в том, что базы данных синхронизированы правильно, то есть все прокси-таблицы обладают правильной схемой содержимого первичной базы данных, которая только что была перезагружена, может понадобиться выполнить инструкцию for proxy_update на сервере, где находится прокси-БД.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду alter database по умолчанию может выполнять только владелец базы данных. Системные администраторы также могут изменять базы данных помощью этой команды.

См. также

Команды create database, disk init, drop database, load database

Системные процедуры sp_addsegment, sp_dropsegment, sp_helpdb, sp_helpsegment, sp_logdevice, sp_renamedb, sp_spaceused

alter role

Описание

Определяет отношения взаимного исключения между ролями; добавляет, удаляет и изменяет пароли для ролей; определяет срок действия пароля, минимальную длину пароля и максимальное количество неудачных попыток подключения к серверу для указанной роли.

Синтаксис

```
alter role роль1 { add | drop } exclusive
        { membership | activation } роль2

alter role имя_роли [add passwd "пароль" |
        drop passwd] [lock | unlock]

alter role { имя_роли | "all overrides" }
        set { passwd expiration | min passwd length |
        max failed_logins } значение_параметра
```

Параметры

роль1
Одна из ролей в паре взаимоисключающих ролей.

add
Добавляет роль, создавая пару ролей, находящихся в отношении взаимного исключения; добавляет пароль к роли.

drop
Удаляет роль из пары ролей, находящихся в отношении взаимного исключения; удаляет пароль из роли.

exclusive
Определяет две указанные роли как взаимоисключающие.

membership
Не позволяет предоставлять пользователям обе роли одновременно.

activation
Позволяет предоставлять пользователю обе роли одновременно, но не разрешает активировать их одновременно.

роль2
Вторая из ролей в паре взаимоисключающих ролей.

имя_роли
Имя роли, пароль которой нужно добавить, удалить или изменить.

passwd
Добавляет пароль к роли.

пароль
Пароль, который нужно добавить к роли. Пароли должны соответствовать правилам для идентификаторов, а их длина должна быть не менее шести символов. Нельзя использовать в качестве паролей переменные.

lock
Блокирует указанную роль.

unlock
Разблокирует указанную роль.

all overrides
Применяет настройку ко всему серверу, а не к определенной роли.

set
Активирует следующий за ним параметр.

passwd expiration
Определяет срок действия пароля в днях. Это может быть любое значение от 0 до 32767 включительно.

min passwd length
Указывает минимальную длину, допустимую для заданного пароля.

max failed_logins
Указывает максимальное количество неудачных попыток подключения к серверу для заданного пароля.

значение_параметра
Указывает значение параметра passwd expiration, min passwd length или max failed_logins. Для активации параметра all overrides переменной *значение_параметра* должно быть присвоено значение -1.

Примеры

Пример 1. Определение ролей intern_role и specialist_role как взаимоисключающих:

```
alter role intern_role add exclusive membership specialist_role
```

Пример 2. Определение ролей как взаимоисключающих на уровне членства (membership) и на уровне активации (activation):

```
alter role specialist_role add exclusive membership intern_role
alter role intern_role add exclusive activation surgeon_role
```

Пример 3. Добавление пароля к существующей роли:

```
alter role doctor_role add passwd "physician"
```

Пример 4. Удаление пароля из существующей роли:

```
alter role doctor_role drop passwd
```

Пример 5. Блокирование роли physician_role:

```
alter role physician_role lock
```

Пример 6. Разблокирование роли physician_role:

```
alter role physician_role unlock
```

Пример 7. Установка для роли physician_role максимального числа неудачных попыток подключения к серверу, равного 5:

```
alter role physician_role set max failed_logins 5
```

Пример 8. Установка минимальной длины пароля (5 символов) для существующей роли physician_role:

```
alter role physician_role set min passwd length 5
```

Пример 9. Снятие ограничения на минимальную длину пароля для всех ролей:

```
alter role "all overrides" set min passwd length -1
```

Пример 10. Снятие ограничения на максимальное количество неудачных попыток подключения к серверу для всех ролей:

```
alter role "all overrides" set max failed_logins -1
```

Использование

- Команда alter role определяет отношения взаимного исключения между ролями и добавляет, удаляет и изменяет пароли ролей.
- Дополнительную информацию об изменении ролей см. в книге *Руководство по системному администрированию*.
- Параметр all overrides удаляет системные настройки, сделанные процедурой sp_configure с любым из следующих параметров:
  - passwd expiration
  - max failed_logins
  - min passwd length

  Когда удаляется пароль для роли, удаляются значения параметров срока действия пароля и максимального количества неудачных попыток подключения к серверу.

Взаимоисключающие роли

- Нет необходимости придерживаться какого-либо определенного порядка при указании ролей, находящихся в отношении взаимного исключения, или в иерархии ролей.
- Можно использовать взаимное исключение вместе с иерархией ролей, чтобы наложить ограничения на пользовательские роли.

- Отношения взаимного исключения на уровне членства (membership) являются более сильным ограничением, чем отношения взаимного исключения на уровне активации (activation). Если две роли определены как взаимоисключающие на уровне членства, то они неявно будут таковыми и на уровне активации.
- Если две роли, уже определенные как взаимоисключающие на уровне членства, будут определены как взаимоисключающие на уровне активации, то это никак не отразится на определениях членства. Отношения взаимного исключения на уровне активации добавляются и удаляются независимо от взаимного исключения на уровне членства.
- Нельзя определить две роли как взаимоисключающие после того, как обе эти роли были предоставлены пользователям или ролям. Необходимо отозвать одну из предоставленных ролей, прежде чем пытаться определить эти роли как взаимоисключающие на уровне членства.
- Если две роли определены как взаимоисключающие на уровне активации, то администратор по безопасности может предоставить обе роли одному пользователю, но эти роли не могут быть одновременно активными у этого пользователя.
- Если администратор по безопасности определяет две роли как взаимоисключающие на уровне активации и пользователи уже активировали обе роли или установили активацию обеих ролей при входе в систему (режим по умолчанию), Adaptive Server делает эти роли взаимоисключающими, но выдает предупреждение с именами соответствующих пользователей с конфликтующими ролями. Активированные роли пользователей не изменяются.

Изменение паролей ролей

- Для изменения пароля роли необходимо сначала удалить существующий пароль, а затем добавить новый, как показано ниже:

```
alter role doctor_role drop passwd
alter role doctor_role add passwd "physician"
```

Примечание. Срок действия паролей, назначенных пользовательским ролям, не ограничен.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду alter role может выполнять только администратор по безопасности системы.

См. также

Команды create role, drop role, grant, revoke, set

Функции mut_excl_roles, proc_role, role_contain, role_id, role_name

Системные процедуры sp_activeroles, sp_displaylogin, sp_displayroles, sp_modifylogin

alter table

Описание

Добавляет новые столбцы к таблице; удаляет или изменяет существующие столбцы; добавляет, изменяет или удаляет ограничения; изменяет свойства существующей таблицы; включает или отключает триггеры для таблицы.

Синтаксис

```
alter table [[база_данных.][владелец].имя_таблицы

    { add имя_столбца тип_данных
        [default {константа | user | null}]
        {identity | null | not null}
        [off row | in row]
        [ [constraint имя_ограничения]
        { { unique | primary key }
            [clustered | nonclustered]
            [asc | desc]
            [with { fillfactor = процент,
                max_rows_per_page = количество_строк,
                reservepagegap = количество_страниц }]
            [on имя_сегмента]
        | references [[база_данных.]владелец.]родительская_таблица
            [(родительский_столбец)]
        | check (условие_поиска) ] ... }
        [, следующий_столбец]...

    | add {[constraint имя_ограничения]
    {  unique | primary key}
            [clustered | nonclustered]
            (имя_столбца [asc | desc]
        [, имя_столбца [asc | desc]...])
        [with { fillfactor = процент,
                max_rows_per_page = количество_строк,
                reservepagegap = количество_страниц}]
        [on имя_сегмента]
    | foreign key (имя_столбца [{, имя_столбца}...])
        | references [[база_данных.]владелец.]родительская_таблица
        [(родительский_столбец [{, родительский_столбец}...])]
    | check (условие_поиска)}

    | drop {имя_столбца [, имя_столбца]...
        | constraint имя_ограничения }

    | modify имя_столбца тип_данных [null | not null]
        [, следующий_столбец]...

    | replace имя_столбца
        default { константа | user | null}
```

| partition *количество_секций*

| unpartition

| { enable | disable } trigger

| lock {allpages | datarows | datapages } }

| with exp_row_size=*количество_байтов*

Параметры

имя_таблицы

Имя таблицы, которую нужно изменить. Если таблица находится в другой базе данных, то должно быть указано имя базы данных. Если в базе данных существует несколько таблиц с таким именем, то должно быть указано имя владельца. По умолчанию *владельцем* является текущий пользователь, а *базой_данных* – текущая база данных.

add

Указывает имя столбца или ограничения, которые нужно добавить в таблицу. При изменении таблиц, находящихся на удаленных серверах, нельзя указывать ключевое слово add, если включены службы Component Integration Services.

имя_столбца

Имя столбца в этой таблице. Если в базе данных можно использовать язык Java, то это может быть столбец Java-SQL.

тип_данных

Любой системный тип данных, кроме bit, или любой определенный пользователем тип данных, кроме основанных на типе bit.

Если в базе данных используется язык Java, то *тип_данных* может быть именем системного или пользовательского класса Java, установленного в базе данных. Дополнительную информацию см. в книге *Java in Adaptive Server Enterprise*.

default

Указывает значение по умолчанию для столбца. Если значение по умолчанию определено, то Adaptive Server вставит его, если пользователь не укажет значение этого столбца при вставке данных. В качестве значения по умолчанию можно использовать *константу*, ключевое слово user (чтобы вставить имя пользователя, который вставляет данные) или ключевое слово null (для вставки значения NULL).

Adaptive Server создает имя для значения по умолчанию в следующей форме: *имя-таблицы_имя-столбца_идентификатор-объекта*, где *имя-таблицы* – это первые десять символов имени таблицы, *имя-*

столбца – первые пять символов имени столбца, а *идентификатор-объекта* – идентификатор объекта для значения по умолчанию. Если задать значению по умолчанию null, то это значение по умолчанию будет удалено.

Параметр default нельзя указывать для удаленных серверов, если включены службы Component Integration Services.

константа

Выражение-константа, которое будет использоваться в качестве значения по умолчанию для столбца. Не должно включать глобальные переменные, имена каких-либо столбцов или других объектов базы данных, но может включать встроенные функции. Это значение по умолчанию должно быть совместимо с типом данных столбца.

user

Указывает на то, что если пользователь не ввел какое-либо значение, в столбец нужно вставить имя пользователя. Столбец должен быть типа char(30), varchar(30) или типа, который Adaptive Server неявно преобразует к типу *char*; однако если столбец имеет тип не char(30) или varchar(30), данные столбца могут быть усечены.

null | not null

Определяет, что произойдет при вставке данных, если значение по умолчанию не задано.

Ключевое слово null указывает, что добавляемый столбец может содержать значения NULL. Если пользователь не ввел значение при вставке, то в этот столбец будет помещено значение NULL.

Ключевое слово not null указывает, что добавляемый столбец не может содержать значения NULL. Если значение по умолчанию не задано, пользователь должен ввести при вставке значение, не равное NULL.

Если не указано ни null, ни not null, то по умолчанию используется установка not null. Впрочем, с помощью системной процедуры sp_dboption можно изменить эту установку по умолчанию, чтобы значение по умолчанию стало совместимым со стандартами SQL. Если для добавляемого столбца указано (или подразумевается) ключевое слово not null, то для него также нужно указать инструкцию default, определяющую значение по умолчанию. Значение по умолчанию будет занесено в добавляемый столбец для всех существующих строк, а также будет использоваться для последующих операций вставки.

identity

Указывает, что столбец обладает свойством IDENTITY. В каждой таблице базы данных может быть только один столбец IDENTITY

типа numeric с масштабом, равным нулю. Столбцы IDENTITY нельзя обновлять. Они не допускают значения NULL.

В столбцах IDENTITY хранятся последовательные номера, например номера счетов-фактур или номера сотрудников, автоматически генерируемые Adaptive Server. Значение в столбце IDENTITY однозначно идентифицирует каждую строку в таблице.

Если включены службы Component Integration Services, то использовать параметр identity для удаленных серверов невозможно.

off row | in row
Указывает, хранится ли столбец Java-SQL вне строки или внутри нее.

Размер столбца, хранящегося внутри строки (in row), не должен превышать 16 КБ (точное значение зависит от размера страницы сервера базы данных и других переменных). Установка по умолчанию – off row.

constraint
Используется для добавления ограничения целостности. Параметр constraint нельзя указывать для удаленных серверов, если включены службы Component Integration Services.

имя_ограничения
Имя ограничения. Оно должно соответствовать правилам для идентификаторов и быть уникальным в базе даных. Если имя для ограничения на уровне таблицы не указано, то Adaptive Server создает имя следующего вида: *имя-таблицы_имя-столбца_идентификатор-объекта*, где *имя-таблицы* – это первые десять символов имени таблицы, *имя-столбца* – первые пять символов имени столбца, а *идентификатор-объекта* – идентификатор объекта для данного ограничения. Если не указано имя для ограничений уникального или первичного ключа, то Adaptive Server создает имя в форме *имя-таблицы_имя-столбца_идентификатор-индекса-таблицы*, где *идентификатор-индекса-таблицы* – строка, полученная конкатенацией идентификатора таблицы и идентификатора индекса.

Ограничения не применяются к данным, уже существующим в таблице на момент добавления ограничения.

unique
Ограничивает значения в указанном столбце или столбцах таким образом, что две строки не могут содержать одинаковое значение, не равное NULL. Это ограничение создает уникальный индекс, удалить который можно только путем удаления ограничения. Нельзя использовать этот параметр с параметром null, описанным выше.

primary key

Ограничивает значения в указанном столбце или столбцах таким образом, что две строки не могут содержать одинаковое значение и ни одна строка не может содержать значение NULL. Это ограничение создает уникальный индекс, удалить который можно только путем удаления ограничения.

clustered | nonclustered

Определяет, кластерным или некластерным является индекс, созданный ограничением unique или primary key. По умолчанию для ограничений по первичному ключу используется значение clustered (если для таблицы еще не создан кластерный индекс), а для ограничений по уникальному ключу – значение nonclustered. Для таблицы может существовать только один кластерный индекс. Дополнительную информацию см. в описании команды create index.

asc | desc

Определяет порядок индексирования: по возрастанию (параметр asc) или по убыванию (параметр desc). По умолчанию индекс имеет порядок по возрастанию.

with fillfactor=процент

Определяет процент заполнения страницы при построении нового индекса по существующим данным. Здесь "*процент*" означает процент заполнения. Настройка fillfactor применяется только при создании индекса. По мере изменения данных процент заполнения страниц не поддерживается на каком-либо определенном уровне.

Предупреждение. Параметр fillfactor, указанный при создании кластерного индекса, влияет на объем, который будут занимать данные, поскольку при создании кластерного индекса происходит перераспределение данных.

По умолчанию значение параметра fillfactor равно 0; оно используется, если в оператор create index не включен параметр with fillfactor (если только это значение не было изменено процедурой sp_configure). В качестве параметра fillfactor можно указывать значение в интервале от 1 до 100.

Если параметр fillfactor равен 0, то создается кластерный индекс с целиком заполненными страницами или некластерный индекс с целиком заполненными листовыми страницами. При этом в B-дереве индекса остается достаточно пространства (в случае как кластерного, так и

некластерного индекса). Необходимость изменять значение параметра fillfactor возникает редко.

Если параметру fillfactor присвоено значение 100, то каждая страница создаваемых кластерных и некластерных индексов будет заполнена на 100 процентов. Значение fillfactor, равное 100, имеет смысл исключительно для таблиц, предназначенных только для чтения, то есть таблиц, в которые никогда не будут добавляться данные.

Если значения fillfactor меньше 100 (кроме 0), новые индексы будут создаваться с частично заполненными страницами. Значение fillfactor, равное 10, может быть оправдано при создании индекса для таблицы, которая со временем будет содержать гораздо больше данных, чем в настоящее время. Однако при малых значениях fillfactor для хранения каждого индекса (или индекса и данных) потребуется больше пространства.

max_rows_per_page=*количество_строк*

Ограничивает количество строк на страницах данных и листовых страницах индексов. В отличие от параметра fillfactor, значение max_rows_per_page ограничивает количество строк на странице до тех пор, пока оно не будет изменено процедурой sp_chgattribute.

Если значение параметра max_rows_per_page не указано, то при создании индекса используется значение 0. Для страниц данных параметр max_rows_per_page принимает значения от 0 до 256. Максимальное количество строк на странице некластерного индекса зависит от размера ключа индекса; если заданное значение слишком велико, Adaptive Server возвращает сообщение об ошибке.

Если индексы создаются ограничениями и параметр max_rows_per_page равен 0, то в кластерных индексах страницы заполняются целиком, а в некластерных – листовые страницы заполняются целиком. Если задано значение 0, то в B-дереве индекса (как кластерного, так и некластерного) остается достаточно пространства.

Если параметру max_rows_per_page присвоено значение 1, то на каждой листовой странице создаваемого кластерного или некластерного индекса будет находиться по одной строке. Это значение можно использовать для уменьшения количества конфликтов блокировок для часто используемых данных.

при небольших значениях параметра max_rows_per_page страницы создаваемых индексов будут заполнены не до конца и будут исполь-

зовать больше пространства для хранения, что может вызвать больше расщеплений страниц.

Предупреждение. Параметр max_rows_per_page, указанный при создании кластерного индекса, влияет на объем пространства, который будут занимать данные, поскольку при создании кластерного индекса происходит перераспределение данных.

---

reservepagegap = количество_страниц

Указывает отношение количества заполненных страниц к количеству страниц, которые будут оставлены пустыми при будущем выделении экстентов под индекс, создаваемый данным ограничением. Этот параметр определяет, на какое *количество_страниц* будет оставлена одна пустая страница для будущего расширения таблицы. Допустимые значения – от 0 до 255. При использовании значения по умолчанию, равного 0, пустые страницы не оставляются.

on имя_сегмента

Указывает, что индекс должен быть создан в заданном сегменте. Перед использованием параметра on имя_сегмента необходимо инициализировать устройство с помощью команды disk init и добавить сегмент к базе данных с помощью системной процедуры sp_addsegment. Список сегментов базы данных можно узнать у системного администратора или с помощью процедуры sp_helpsegment.

Если одновременно указаны параметры clustered и on имя_сегмента, вся таблица перемещается в указанный сегмент, так как листовой уровень индекса содержит страницы с данными.

references

Указывает список столбцов для ограничения ссылочной целостности. Можно указать только одно значение столбца для ограничения на уровне столбца. Если это ограничение включено в таблицу, ссылающуюся на другую таблицу, то любые данные, которые вставляются в *дочернюю* таблицу, должны существовать в *родительской* таблице.

Для использования этого ограничения необходимо обладать полномочиями references для родительской таблицы. Указанные столбцы в родительской таблице должны быть ограничены уникальным индексом (созданным ограничением unique или оператором create index). Если столбцы не указаны, то в родительской таблице должно быть

определено ограничение primary key на соответствующих столбцах. Кроме того, типы данных столбцов дочерней таблицы должны точно совпадать с типами данных столбцов родительской таблицы.

Параметр references нельзя указывать для удаленных серверов, если включены службы Component Integration Services.

foreign key
Указывает, что перечисленные столбцы являются внешними ключами в этой таблице и соответствуют первичным ключам, указанным в инструкции references.

родительская_таблица
Имя таблицы, содержащей столбцы, на которые ссылается внешний ключ. Родительская таблица может находиться в другой базе данных. Ограничения могут ссылаться не более чем на 192 пользовательские таблицы и рабочие таблицы, созданные системой. Для проверки ссылочных ограничений таблицы используется системная процедура sp_helpconstraint.

родительский_столбец
Имя столбца или столбцов в родительской таблице.

check
Указывает *условие_поиска*, которое проверяется для всех строк в таблице. Если включены службы Component Integration Services, то параметр check нельзя использовать для удаленных серверов.

условие_поиска
Логическое выражение, которое определяет ограничение check для значений столбца. Эти ограничения могут включать:

- список выражений-констант, предваряемый ключевым словом in;
- набор условий, в котором могут присутствовать метасимволы (предваряемый ключевым словом like).

  Выражение может содержать арифметические операции и функции Transact-SQL, но *условие_поиска* не может содержать подзапросы, агрегатные функции, параметры и переменные базового языка.

следующий_столбец
Включает дополнительные определения столбцов (через запятую), используя тот же синтаксис, что и описанный выше синтаксис определения столбца.

drop
: Указывает имя столбца или ограничения, которые нужно удалить из таблицы. Параметр drop нельзя указывать для удаленных серверов, если включены службы Component Integration Services.

modify
: Указывает имя столбца, для которого нужно изменить тип данных или допустимость значений NULL.

replace
: Указывает столбец, для которого нужно изменить значение по умолчанию на новое, которое задано в следующей инструкции default. Если включены службы Component Integration Services, то параметр replace нельзя использовать для удаленных серверов.

partition количество_секций
: Создает несколько цепочек страниц базы данных для таблицы. Операции вставки могут одновременно выполняться на последних страницах всех цепочек. Переменная *количество_секций* должна принимать положительные целочисленные значения, большие или равные 2. Для каждой секции необходима дополнительная служебная страница; число секций, которые можно создать в таблице, может быть ограничено дисковым пространством. Число секционированных таблиц, к которым можно обратиться, также ограничено памятью. Если включены службы Component Integration Services, то параметр partition нельзя использовать для удаленных серверов.

unpartition
: Создает одну цепочку страниц для таблицы путем присоединения к первой цепочке последующих. Если включены службы Component Integration Services, то параметр unpartition нельзя указывать для удаленных серверов.

enable | disable trigger
: Включает или отключает триггер. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

lock datarows | datapages | allpages
: Изменяет схему блокировки для таблицы.

with exp_row_size=количество_байтов

Указывает ожидаемый размер строки. Применяется только к схемам блокировки строк данных и страниц данных, к таблицам со строками переменной длины и только при копировании данных командой alter table. Допустимыми значениями являются 0, 1 и любое значение, лежащее в интервале между минимальной и максимальной длиной строки в таблице. По умолчанию используется значение 0, которое означает, что применяется установка, заданная на уровне сервера.

Примеры

Пример 1. Добавление столбца к таблице. В этот столбец заносится значение NULL для каждой существующей строки в этой таблице:

```
alter table publishers
add manager_name varchar(40) null
```

Пример 2. Добавление к таблице столбца IDENTITY. Для всех существующих строк этой таблицы в этот столбец будут занесены уникальные последовательные значения. Следует обратить внимание на то, что столбец IDENTITY имеет тип numeric и масштаб 0. Точность определяет максимальное значение, которое может быть записано в столбец (это максимальное значение равно $10^5 - 1$, или 99 999):

```
alter table sales_daily
add ord_num numeric(5,0) identity
```

Пример 3. Добавление ограничения первичного ключа в таблицу authors. Если ограничение первичного ключа или ограничение unique уже существует, то необходимо сначала удалить его (см. пример 5):

```
alter table authors
add constraint au_identification
primary key (au_id, au_lname, au_fname)
```

Пример 4. Создание индекса для таблицы authors. Для этого индекса параметр reservepagegap равен 16, поэтому на каждые 15 выделенных страниц в индексе будет оставлена одна пустая страница:

```
alter table authors
add constraint au_identification
primary key (au_id, au_lname, au_fname)
with reservepagegap = 16
```

Пример 5. Удаление ограничения au_identification:

```
alter table titles
drop constraint au_identification
```

Пример 6. Удаление значения по умолчанию для столбца phone таблицы authors. Если пользователь не указывает значение столбца, который допускает неопределенные значения NULL, то в столбец вставляется NULL. Если столбец не допускает значения NULL, то выполнить вставку, не указывая значение для столбца, не удастся:

```
alter table authors
replace phone default null
```

Пример 7. Создание четырех новых цепочек страниц в таблице titleauthor (то есть разделение таблицы на 5 секций). После секционирования таблицы существующие данные остаются в первой секции. Но новые строки будут вставляться во все пять секций:

```
alter table titleauthor partition 5
```

Пример 8. Объединение всех цепочек страниц таблицы titleauthor, а затем ее повторное разделение на шесть секций:

```
alter table titleauthor unpartition
alter table titleauthor partition 6
```

Пример 9. Изменение текущей схемы блокировки таблицы titles на схему блокировки строк данных:

```
alter table titles lock datarows
```

Пример 10. Добавление в таблицу authors столбца author_type, не допускающего значений NULL и имеющего значение по умолчанию primary_author:

```
alter table authors
    add author_type varchar(20)
    default "primary_author" not null
```

Пример 11. Удаление столбцов advance, notes и contract из таблицы titles:

```
alter table titles
    drop advance, notes, contract
```

Пример 12. Изменение столбца city таблицы authors. Текущий тип данных заменяется на тип varchar(30) со значением по умолчанию, равным NULL:

```
alter table authors
    modify city varchar(30) null
```

Пример 13. Указание для столбца stor_name таблицы stores свойства NOT NULL. При этом тип данных столбца (varchar(40)) не меняется:

```
alter table stores
    modify stor_name not null
```

Пример 14. Изменение столбца type таблицы titles, изменение схемы блокировки таблицы titles с блокировки всех страниц на блокировку строк данных:

```
alter table titles
    modify type varchar(10)
    lock datarows
```

Пример 15. Изменение типа данных столбца notes таблицы titles (с varchar(200) на varchar(150)) и допустимости значений NULL (с NULL на NOT NULL), а также установка параметра exp_row_size в 40:

```
alter table titles
    modify notes varchar(150) not null
    with exp_row_size = 40
```

Пример 16. Добавление, изменение и удаление столбцов, а затем добавление еще одного столбца в одном операторе. Кроме того, этот оператор также изменяет схему блокировки и задает значение параметра exp_row_size для нового столбца:

```
alter table titles
    add author_type varchar(30) null
    modify city varchar(30)
    drop notes
    add sec_advance money default 1000 not null
    lock datarows
    with exp_row_size = 40
```

Использование

- Если хранимая процедура содержит оператор select *, делающий выборку из таблицы, структура которой была изменена, то в результирующем наборе новые столбцы не появятся (даже если процедура была создана с параметром with recompile). Необходимо удалить процедуру и создать ее заново, чтобы включить эти новые столбцы.
- Если команда alter table выполняется владельцем таблицы, правила доступа отключаются на время выполнения этой команды (снова включаясь после ее завершения). Это делается для того, чтобы не фильтровать данные таблицы во время выполнения команды alter table.

Ограничения

Предупреждение. Системные таблицы изменять нельзя.

- Нельзя добавлять столбец типа bit в существующую таблицу.

- Максимальное число столбцов в таблице определяется следующим образом:
  - 1024 для столбцов фиксированной длины в таблицах как с блокировкой всех страниц (all-pages-locked, APL), так и с блокировкой только данных (data-only-locked, DOL);
  - 254 для столбцов переменной длины в таблицах с блокировкой всех страниц;
  - 1024 для столбцов переменной длины в таблицах с блокировкой только данных.
- Если в команде alter table число столбцов переменной длины в таблице с блокировкой всех страниц превысит 254, будет выдано сообщение об ошибке.
- Максимальная длина столбца Java, хранящегося в строке, определяется максимальным размером столбца переменной длины для схемы таблицы, типом блокирования и размером страницы.
- При изменении схемы блокировки для таблицы данные исходной таблицы не должны нарушать ограничения, налагаемые на целевую таблицу. Например, при попытке преобразовать таблицу с блокировкой только данных (DOL), в которой более 254 столбцов переменной длины, в таблицу с блокировкой всех страниц (APL) команда alter table даст сбой, поскольку число столбцов переменной длины в таблице APL не должно превышать 254.
- Максимальные размеры столбцов, содержащих данные фиксированной длины (например, столбцов типа char, binary и др.), приведены в следующей таблице:

Таблица 7-2. Максимальная длина строки и столбца в таблицах APL и DOL

Схема блокировки	Размер страницы	Максимальная длина строки	Максимальная длина столбца
Таблицы APL	2 КБ (2048 байтов)	1962	1960 байтов
	4 КБ (4096 байтов)	4010	4008 байтов
	8 КБ (8192 байта)	8106	8104 байта
	16 КБ (16384 байта)	16298	16296 байтов
Таблицы DOL	2 КБ (2048 байтов)	1964	1958 байтов
	4 КБ (4096 байтов)	4012	4006 байтов
	8 КБ (8192 байта)	8108	8102 байта

Схема блокировки	Размер страницы	Максимальная длина строки	Максимальная длина столбца
	16 КБ (16384 байта)	16300	16294 байта при отсутствии в таблице столбцов переменной длины
	16 КБ (16384 байта)	16300 (при максимальном начальном смещении = 8191)	8191-6-2 = 8183 байта при наличии в таблице хотя бы одного столбца переменной длины*

* В этот размер включены шесть байтов для служебной части строки и два байта для поля длины строки.

- Максимальный объем (в байтах) данных переменной длины в строке зависит от схемы блокировки для таблицы. Ниже приведены максимальные размеры столбцов для таблиц APL:

Размер страницы	Максимальная длина строки	Максимальная длина столбца
2 КБ (2048 байтов)	1960	1960
4 КБ (4096 байтов)	4008	4008
8 КБ (8192 байта)	8104	8157
16 КБ (16384 байта)	16296	16227

  Ниже приведены максимальные размеры столбцов для таблиц DOL:

Размер страницы	Максимальная длина строки	Максимальная длина столбца
2 КБ (2048 байтов)	1960	1958
4 КБ (4096 байтов)	4008	4006
8 КБ (8192 байта)	8157	8102
16 КБ (16384 байта)	16294	16294

- Нельзя секционировать системную таблицу или таблицу, которая уже секционирована.
- Команду alter table с инструкцией partition или unpartition нельзя выполнять в пользовательской транзакции.
- Максимальное значение параметра max_rows_per_page для таблиц APL составляет 256 байтов. Для таблиц DOL параметр max_rows_per_page не используется.

- В одном пакете или процедуре нельзя сначала добавить декларативное ограничение или ограничение check командой alter table, а затем вставить данные в эту таблицу. Необходимо либо поместить операторы изменения и вставки в разные пакеты или процедуры, либо использовать команду execute для раздельного выполнения этих действий.
- В команде alter table нельзя использовать переменные для указания значений по умолчанию:

```
declare @a int
select @a = 2
alter table t2 add c3 int
default @a
```

  Если сделать это, будет выдано сообщение об ошибке 154: "Variable is not allowed in default".

Получение сведений о таблицах

- Информацию о таблице и ее столбцах можно получить с помощью процедуры sp_help.
- Для переименования таблицы используется системная процедура sp_rename (системные таблицы переименовывать нельзя).
- Дополнительную информацию об ограничениях целостности (unique, primary key, references и check) и инструкции default см. в описании команды create table в этой главе.

Указание порядка сортировки в индексах

- Порядок сортировки для индекса задается с помощью ключевых слов asc и desc, которые указываются после имен столбцов индекса. Если в инструкции order by запроса столбцы указаны в том же порядке, в котором они были указаны при создании какого-либо существующего индекса, то при обработке этого запроса можно будет обойтись без сортировки. Дополнительную информацию см. в главе 8, "Индексирование для повышения производительности", книги *Руководство по настройке производительности*.

Использование межбазовых ограничений ссылочной целостности

- Если создается межбазовое ограничение (то есть ограничение, связывающее таблицы, расположенные в разных базах данных), в системных таблицах sysreferences обеих баз данных сохраняется следующая информация:

Таблица 7-3. Информация об ограничениях ссылочной целостности, хранящаяся в системных таблицах

Информация, хранящаяся в таблице sysreferences	Столбцы с информацией о родительской таблице	Столбцы с информацией о дочерней таблице
Идентификаторы ключевого столбца	с refkey1 по refkey16	с fokey1 по fokey16
Идентификатор таблицы	reftabid	tableid
Идентификатор базы данных	pmrydbid	frgndbid
Имя базы данных	pmrydbname	frgndbname

- При удалении дочерней таблицы или базы данных, в которой она находится, Adaptive Server удаляет сведения о внешнем ключе из базы данных, на таблицу которой ссылалась дочерняя таблица.
- Поскольку дочерняя таблица зависит от информации из таблицы, на которую она ссылается, нельзя выполнять следующие действия:
  - удалять родительскую таблицу;
  - удалять внешнюю базу данных, содержащую родительскую таблицу;
  - переименовывать какую-либо из баз данных, участвующих в ограничении, с помощью процедуры sp_renamedb.

  Сначала необходимо удалить межбазовое ограничение с помощью команды alter table.
- Каждый раз при добавлении или удалении межбазового ограничения или при удалении таблицы, содержащей такое ограничение, необходимо делать резервные копии *обеих* баз данных.

  Предупреждение. Загрузка более ранних резервных копий может привести к повреждению этих баз данных.
- Имя и идентификатор внешней базы данных хранятся в системной таблице sysreferences. Adaptive Server не может гарантировать ссылочную целостность при выполнении команды load database,

переименовывающей базу данных или загружающей ее на другой сервер.

Предупреждение. Прежде чем создать резервную копию базы данных для того, чтобы затем загрузить ее под другим именем или переместить на другой сервер Adaptive Server, необходимо удалить все внешние ограничения ссылочной целостности с помощью команды alter table.

Изменение значений по умолчанию

- Значения по умолчанию для столбца можно задать двумя способами: либо объявить значение по умолчанию как ограничение для столбца в операторах create table или alter table, либо создать его оператором create default и назначить это значение столбцу процедурой sp_bindefault.
- Нельзя заменить определенное пользователем значение по умолчанию, назначенное столбцу с помощью процедуры sp_bindefault. Сначала необходимо отменить назначение процедурой sp_unbindefault.
- Если значение по умолчанию для столбца объявлено с помощью create table или alter table, то назначить этому столбцу другое значение по умолчанию с помощью sp_bindefault нельзя. Сначала необходимо удалить значение по умолчанию, заменив его на NULL, а затем назначить значение по умолчанию, определенное пользователем. Когда значение по умолчанию изменяется на NULL, назначение этого значения отменяется и оно удаляется из таблицы sysobjects.

Настройка параметров управления пространством для индексов

- Параметры управления пространством fillfactor, max_rows_per_page и reservepagegap в операторе alter table применяются к индексам, созданным для ограничений primary key или unique. Параметры управления пространством влияют на страницы данных таблицы, если это ограничение создает кластерный индекс для таблицы с блокировкой всех страниц.
- Для изменения значения параметров max_rows_per_page или reservepagegap для таблицы или индекса, изменения значения параметра exp_row_size для таблицы или сохранения значения fillfactor используется процедура sp_chgattribute.
- Параметры управления пространством для индексов применяются в следующих случаях:

- При повторном создании индексов в результате выполнения команды alter table, которая изменяет схему блокировки таблицы с блокировки всех страниц на блокировку только данных или наоборот. Дополнительную информацию см. в разделе “Изменение схем блокировки” на стр. 310.
- При автоматическом перестроении индексов в ходе выполнения команды reorg rebuild.

- Для просмотра текущих параметров управления пространством, действующих в отношении таблицы, используется процедура sp_help. Для просмотра текущих параметров управления пространством, действующих в отношении индекса, используется процедура sp_helpindex.
- Параметры управления пространством fillfactor, max_rows_per_page и reservepagegap помогают управлять использованием пространства для таблиц и индексов следующим образом:
  - Параметр fillfactor оставляет дополнительное пространство на страницах при создании индексов, однако после создания индекса размер свободного пространства больше не привязан к значению параметра fillfactor и может изменяться. Этот параметр применим ко всем схемам блокировки.
  - Параметр max_rows_per_page ограничивает количество строк на странице данных или странице индекса. В основном он используется, чтобы улучшить одновременный доступ к таблицам с блокировкой всех страниц.
  - Параметр reservepagegap указывает соотношение пустых страниц и заполненных для команд, выделяющих экстенты. Параметр применим ко всем схемам блокировки.

  Параметры управления пространством сохраняются для таблиц и индексов и применяются в ходе выполнения команд alter table и reorg rebuild.
- Допустимые сочетания параметров управления пространством и схем блокировки перечислены в следующей таблице. Если команда alter table изменяет таблицу таким образом, что сочетание параметра и схемы блокировки становится недопустимым, то данные в системных таблицах остаются, но во время операций над таблицей не применяются. Если схема блокировки таблицы изменяется так, что параметры становятся допустимыми, то они применяются.

Параметр	Блокировка всех страниц	Блокировка страниц данных	Блокировка строк данных
max_rows_per_page	Да	Нет	Нет
reservepagegap	Да	Да	Да
fillfactor	Да	Да	Да
exp_row_size	Нет	Да	Да

- В следующей таблице перечислены значения по умолчанию и описан результат их использования для параметров управления пространством:

Параметр	Значение по умолчанию	Результат использования значения по умолчанию
max_rows_per_page	0	Помещает на страницу максимально возможное количество строк, до 255
reservepagegap	0	Не оставляет интервалов между страницами
fillfactor	0	Целиком заполняет листовые страницы

Преобразование *max_rows_per_page* в *exp_row_size*

- Если таблица, для которой задано значение max_rows_per_page, преобразуется из режима блокировки всех страниц в режим блокировки только данных, то значение этого параметра преобразуется в значение exp_row_size перед тем, как команда alter table...lock скопирует эту таблицу в новое местоположение. Значение параметра exp_row_size применяется в ходе копирования. Преобразование значений этих параметров описано в следующей таблице.

Значение max_rows_per_page	Значение exp_row_size
0	Значение в процентах, заданное командой default exp_row_size percent
255	1, то есть целиком заполненные страницы
1 – 254	Меньшее из следующих значений: • максимальный размер строки; • 2002/значение параметра max_rows_per_page.

Использование параметра *reservepagegap*

- Команды, которым требуется большой объем пространства, выделяют пространство экстентами, а не отдельными страницами. Если использовано ключевое слово reservepagegap, то эти команды остав-

ляют пустые страницы, благодаря чему страницы, которые будут выделены в будущем, окажутся ближе к расщепляемой странице или странице, с которой перемещается строка.

- Значение параметра reservepagegap для таблицы хранится в таблице sysindexes и применяется при изменении схемы блокировки для таблицы с блокировки всех страниц на блокировку только данных и наоборот. Чтобы изменить это значение, перед командой alter table нужно выполнить системную процедуру sp_chgattribute.
- Значение reservepagegap, указанное с ключевым словом clustered для таблицы с блокировкой всех страниц, заменяет любое значение, заданное ранее с помощью команд create table или alter table.

Секционирование таблиц для ускорения операций вставки

- При секционировании таблицы инструкцией partition команды alter table создаются дополнительные цепочки страниц, что позволяет выполнять вставку на последние страницы одновременно. Это ускоряет операции вставки за счет сокращения конкуренции за доступ к страницам, а также за счет сокращения конфликтов между операциями ввода-вывода при записи данных из кэша на диск (если сегмент, содержащий таблицу, расположен на нескольких физических устройствах).
- Чтобы можно было копировать данные в секционированную таблицу или из нее, необходимо сконфигурировать Adaptive Server для параллельной обработки.
- Когда происходит секционирование таблицы, для каждой секции (в том числе и для первой) выделяется служебная страница. Существующая цепочка страниц становится частью первой секции, а для каждой последующей секции создается первая страница. Секционированные таблицы требуют немного больше места на диске, чем несекционированные, поскольку каждая секция имеет свою служебную страницу.
- Секционировать можно как пустые таблицы, так и таблицы, содержащие данные. При секционировании таблицы данные *не* перемещаются; существующие данные остаются в исходном месте хранения (в первой секции). Чтобы увеличить производительность, таблицу нужно секционировать *перед* вставкой данных.
- Нельзя секционировать системную таблицу или таблицу, которая уже секционирована. Можно секционировать таблицу, содержащую столбцы типа text и image, однако это не повлияет на хранение этих столбцов.

- После секционирования таблицы к ней нельзя применять команду truncate table и системную процедуру sp_placeobject.
- Для изменения количества секций в таблице нужно сначала десекционировать таблицу (соединить все существующие цепочки страниц), используя команду alter table с инструкцией unpartition, а затем повторно секционировать таблицу с помощью команды alter table с инструкцией partition.
- После десекционирования таблицы необходимо перекомпилировать планы запросов всех зависимых процедур. При десекционировании перекомпиляция процедур не происходит автоматически.
- Когда таблица десекционируется (командой alter table с инструкцией unpartition), все служебные страницы, включая служебную страницу для первой секции, освобождаются и цепочки страниц соединяются в одну. В получившейся цепочке страниц не будет пустых страниц, кроме, возможно, первой. При десекционировании таблицы ее данные *не* перемещаются.

Добавление столбцов IDENTITY

- При добавлении в таблицу столбца IDENTITY необходимо убедиться, что точность этого столбца достаточна для представления количества существующих строк. Если количество строк превышает величину $10^{\text{точность}} - 1$, СУБД Adaptive Server выдает сообщение об ошибке и не добавляет столбец.
- При добавлении в таблицу столбца IDENTITY СУБД Adaptive Server выполняет следующие операции:
  - Блокирует таблицу, пока не будут сгенерированы все значения столбца IDENTITY. Если таблица содержит большое количество строк, то этот процесс может занять много времени.
  - Назначает каждой существующей строке уникальное последовательное значение столбца IDENTITY, начиная с 1.
  - Вносит в журнал каждую операцию вставки в таблицу. Перед добавлением столбца IDENTITY в таблицу с большим количеством строк используется команда dump transaction, чтобы очистить журнал транзакций базы данных.
- При каждом добавлении строки в таблицу СУБД Adaptive Server генерирует значение столбца IDENTITY, которое на единицу больше предыдущего. Это значение имеет приоритет над любыми значениями по умолчанию, объявленными для этого столбца в операторе alter table или назначенными ему с помощью процедуры sp_bindefault.

Изменение схемы таблицы

- Для изменения схемы существующей таблицы применяются подинструкции add, drop или modify и lock. Один оператор может содержать любое число этих подинструкций в любом порядке, если только имя одного и того же столбца не будет упоминаться более одного раза.
- Если хранимая процедура содержит оператор select *, делающий выборку из таблицы, структура которой была изменена, то в результирующем наборе новые столбцы не появятся (даже если эта процедура была создана с параметром with recompile). Необходимо удалить такую процедуру и создать ее заново, чтобы включить эти новые столбцы.
- Нельзя удалить все столбцы таблицы. Также нельзя удалить из таблицы последний оставшийся столбец (например, если из таблицы, содержащей 5 столбцов, удалить 4 столбца, то удалить оставшийся столбец нельзя). Для удаления таблицы из базы данных используется команда drop table.
- Копирование данных требуется в следующих случаях:
  - для удаления столбца;
  - для добавления столбца, который не должен содержать значения NULL;
  - для большинства команд alter table ... modify.

  Определить, требуется ли копирование данных для конкретной команды alter table, можно, используя параметр showplan.
- Можно внести изменения в схему блокировки таблицы, если при ее изменении с помощью команды alter table (add, drop или modify), потребуется копирование данных.
- Если команда alter table копирует данные, то в базе данных, содержащей таблицу, схему которой нужно изменить, должен быть включен параметр select into /bulkcopy/pllsort.
- При изменении схемы секционированной таблицы, которое требует копирования данных, необходимо настроить СУБД Adaptive Server для параллельной обработки.
- Измененная таблица сохраняет существующие параметры управления пространством (max_rows_per_page, fillfactor и т. п.) и индексы.
- Команда alter table, которая требует копирования данных, не вызывает срабатывания триггеров.

- Для изменения схемы удаленной прокси-таблицы, созданной и поддерживаемой службами Component Integration Services (CIS), используется команда alter table. Дополнительную информацию о CIS см. в книге *Component Integration Services User's Guide*.
- В рамках одного оператора нельзя выполнить копирование данных и добавить ограничение на уровне таблицы или ограничение ссылочной целостности.
- В рамках одного оператора нельзя выполнить копирование данных и создать кластерный индекс.
- При добавлении столбца, не допускающего значения NULL, необходимо указать значение по умолчанию в инструкции default. Из этого правила есть одно исключение: при добавлении столбца определенного пользователем типа не требуется указывать инструкцию default, если с этим типом связано значение по умолчанию.
- В таблицах с блокировкой всех страниц всегда можно добавить, удалить или изменить столбец. Однако в таблицах с блокировкой только данных существуют ограничения на добавление, удаление или изменение столбца, которые описаны в следующей таблице.

Тип индекса	Секционированная таблица с блокировкой всех страниц	Несекционированная таблица с блокировкой всех страниц	Секционированная таблица с блокировкой только данных	Несекционированная таблица с блокировкой только данных
Кластерный	Да	Да	Нет	Да
Некластерный	Да	Да	Да	Да

Если необходимо добавить, удалить или изменить столбец в секционированной таблице с кластерным индексом и блокировкой только данных, можно выполнить следующие шаги:

a удалить кластерный индекс;

b изменить таблицу (с блокировкой только данных);

c заново создать кластерный индекс.

- Нельзя добавлять в качестве столбца объект Java, который не допускает значения NULL. По умолчанию все столбцы Java всегда имеют значение по умолчанию NULL и хранятся как строки типа varbinary или как данные типа image.

- Нельзя изменить секционированную таблицу, содержащую столбец Java, если такое изменение требует копирования данных. Сначала нужно десекционировать секционированную таблицу, затем выполнить команду alter table и повторно секционировать таблицу.
- Нельзя удалить ключевой столбец из индекса или ограничения ссылочной целостности. Для удаления ключевого столбца необходимо сначала удалить индекс или ограничение ссылочной целостности, а затем ключевой столбец. Дополнительную информацию см. в книге *Transact-SQL User's Guide*.
- Можно удалить столбцы, с которыми связаны значения по умолчанию или правила. При удалении столбца также удаляются все значения по умолчанию, относящиеся к этому столбцу. Нельзя удалить столбцы, с которыми связаны проверочные ограничения check или ограничения ссылочной целостности. Сначала необходимо удалить эти ограничения, а затем – столбец. Проверить наличие каких-либо ограничений для таблицы можно с помощью системной процедуры sp_helpconstraint. Для идентификации всех зависимостей уровня столбцов используется процедура sp_depends.
- Нельзя удалить столбец из системной таблицы. Также нельзя удалить столбцы из пользовательских таблиц, которые созданы и используются хранимыми процедурами и стандартными инструментами Sybase.
- В большинстве случаев можно изменить тип данных существующего столбца, если таблица не содержит данных. Если таблица содержит данные, то можно изменить тип данных на любой другой тип, который можно явно преобразовать к исходному типу.
- Со столбцами IDENTITY можно выполнять следующие действия:
  - добавлять новый столбец IDENTITY;
  - удалять существующий столбец IDENTITY;
  - изменять размер существующего столбца IDENTITY.

  Дополнительную информацию см. в книге *Transact-SQL User's Guide*.
- При изменении схемы таблицы увеличивается счетчик ее схемы, в результате чего существующие хранимые процедуры, обращающиеся к этой таблице, повторно нормализуются при их очередном выполнении. Изменения в хранимых процедурах или представлениях, зависящих от типов данных, могут привести к ошибкам нормализации типов данных. Необходимо обновить эти зависимые объекты так, чтобы они ссылались на измененную схему таблицы.

Ограничения на изменение схемы таблицы

- Нельзя выполнять команду alter table в транзакции.
- При изменении схемы таблицы резервные копии, которые были созданы с помощью команды bcp, могут стать недействительными. В таких резервных копиях может использоваться схема таблицы, несовместимая с ее текущей схемой.
- Можно добавлять столбцы NOT NULL с ограничениями check, однако существующие данные не будут проверяться на соответствие этому ограничению.
- Нельзя изменять схему блокировки таблицы командами alter table . . . add, drop или modify, если по этой таблице построен кластерный индекс и операция требует копирования данных. Вместо этого можно выполнить следующие шаги:

  a удалить кластерный индекс;

  b изменить схему таблицы;

  c заново создать кластерный индекс.

- Нельзя изменить схему таблицы, если эту таблицу использует какой-либо активный открытый курсор.

Ограничения на изменение столбцов типа *text* и *image*

- Можно добавлять только те столбцы типа text или image, которые допускают значения NULL.

  Чтобы добавить столбец типа text или image, содержащий только значения, не равные NULL, нужно сначала добавить столбец, допускающий неопределенные значения, а затем обновить его, чтобы он не принимал неопределенные значения.

- Если столбец имеет тип данных text, то изменить этот тип можно только на один из следующих:
  - char
  - varchar
  - unichar
  - univarchar
  - nchar
  - nvarchar
- Столбец типа image можно преобразовать только к типу varbinary, при этом столбец не должен содержать значений NULL.

- Изменить тип столбцов text или image на любой другой можно только в том случае, если таблица пуста.
- В одном операторе нельзя добавить новый столбец типа text или image и удалить существующий столбец типа text или image.
- Нельзя изменить тип данных какого-либо столбца на text или image.

Изменение схем блокировки

- Команда alter table позволяет изменить любую схему блокировки на любую другую. Можно произвести следующие изменения схемы:
  - с allpages (блокировка всех страниц) на datapages (блокировка страниц данных) и наоборот;
  - с allpages (блокировка всех страниц) на datarows (блокировка строк данных) и наоборот;
  - с datapages (блокировка страниц данных) на datarows (блокировка строк данных) и наоборот.
- Прежде чем изменять схему блокировки с блокировки всех страниц на блокировку только данных или наоборот, необходимо присвоить параметру базы данных select into/bulkcopy/pllsort значение true при помощи процедуры sp_dboption, а затем выполнить в базе данных команду checkpoint, если хотя бы одна из таблиц секционирована и для сортировки индексов требуется параллельная сортировка.
- После изменения схемы с блокировки всех страниц на блокировку только данных и наоборот использование команды dump transaction для резервного копирования журнала транзакций запрещено; сначала необходимо сделать полную резервную копию базы данных.
- При использовании команды alter table...lock для изменения схемы блокировки таблицы с блокировки всех страниц на блокировку только данных и наоборот СУБД Adaptive Server делает копию страниц данных этой таблицы. В сегменте, где размещается таблица, должно быть достаточно пространства для полной копии страниц данных. В сегменте, где размещаются индексы, должно быть место для перестроения индексов.

  В кластерных индексах для таблиц с блокировкой только данных уровень листьев находится выше страниц данных. При изменении схемы таблицы с кластерным индексом с блокировки всех страниц на блокировку только данных результирующий кластерный индекс займет больше места. Объем требуемого дополнительного пространства зависит от размера индексных ключей.

С помощью процедуры sp_spaceused можно определить, какое пространство занимает таблица в настоящий момент, а с помощью процедуры sp_helpsegment – какое пространство доступно для хранения таблицы.

- При изменении схемы блокировки таблицы с блокировки всех страниц на блокировку только страниц данных и наоборот параметры управления пространством применяются к таблице во время копирования строк данных и к индексам, когда они воссоздаются. При изменении одной схемы блокировки только данных на другую страницы данных не копируются и параметры управления пространством не применяются.
- Если таблица секционирована, то изменение схемы блокировки приводит к копированию строк из одной секции в другую. В ходе копирования данные не распределяются по секциям равномерно.
- При изменении схемы блокировки таблицы команда alter table...lock устанавливает монопольную блокировку таблицы до завершения своего выполнения.
- При использовании alter table...lock для перехода от блокировки страниц данных к блокировке строк данных страницы данных не копируются и индексы не перестраиваются. Происходит только обновление системных таблиц.
- Если во время изменения схемы блокировки таблицы в системе были другие активные пользователи, то это может повлиять на их действия следующим образом:
  - планы запросов в процедурном кэше, которые обращаются к этой таблице, будут перекомпилированы при их следующем запуске;
  - активные многооператорные процедуры, которые используют эту таблицу, перекомпилируются перед выполнением следующего шага;
  - специальные пакетные транзакции, использующие эту таблицу, завершаются.

Предупреждение. Изменение схемы блокировки таблицы в ходе операции массового копирования может привести к повреждению таблицы. Операция массового копирования сначала получает сведения о таблице и не удерживает блокировку между моментом считывания информации о таблице и началом пересылки строк, оставляя небольшой промежуток времени, когда может быть запущена команда alter table...lock.

Добавление столбцов Java-SQL

- Если в базе данных используется язык Java, то в таблицу можно добавлять столбцы Java-SQL. Дополнительную информацию см. в книге *Java in Adaptive Server Enterprise*.
- Объявленный класс (*тип_данных*) нового столбца Java-SQL должен реализовывать интерфейс Serializable или Externalizable.
- Ограничения на добавление столбца Java-SQL:
  - этот столбец нельзя сделать внешним ключом;
  - этот столбец нельзя указывать в инструкции references;
  - для этого столбца нельзя задать свойство UNIQUE;
  - этот столбец нельзя сделать первичным ключом.
- Если указан параметр in row (то есть столбец будет храниться внутри строки), то хранимое в нем значение не должно превышать 16 КБ (точный предел зависит от размера страницы сервера данных).
- Если указан параметр off row, то:
  - на этот столбец нельзя ссылаться в ограничении check;
  - этот столбец нельзя указывать в операторе select с инструкцией distinct;
  - этот столбец нельзя указывать в операторе сравнения, в предикате и в инструкции group by.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Информацию о совместимости типов данных см. в главе 1, “Системные и пользовательские типы данных”.

Полномочия

Команду alter table по умолчанию может использовать только владелец таблицы. Эти полномочия нельзя передать другому пользователю, кроме владельца базы данных, который работает от лица владельца таблицы с помощью команды setuser. Системный администратор также может изменять пользовательские таблицы командой alter table.

См. также

Команды create index, create table, dbcc, drop database, dump transaction, insert, setuser

Системные процедуры sp_chgattribute, sp_help, sp_helpartition, sp_rename

begin...end

Описание

Определяет блок операторов SQL, чтобы такие управляющие конструкции, как if...else, влияли на выполнение всего блока операторов.

Синтаксис

```
begin
        блок операторов
end
```

Параметры

блок операторов

Последовательность операторов, заключенных между командами begin и end.

Примеры

Пример 1. Использование условия if без команд begin и end приведет к выполнению только одного оператора SQL:

```
if (select avg(price) from titles) < $15
begin
    update titles
    set price = price * $2
    select title, price
    from titles
    where price > $28
end
```

Пример 2. Если бы не использовались команды begin и end, оператор print не выполнился бы:

```
create trigger deltitle
on titles
for delete
as
if (select count(*) from deleted, salesdetail
 where salesdetail.title_id = deleted.title_id) > 0
    begin
        rollback transaction
        print "You can't delete a title with sales."
    end
else
    print "Deletion successful--no sales for this
        title."
```

Использование

- Блоки begin...end можно вкладывать в другие блоки begin...end.

Стандарты	Уровень соответствия стандарту SQL92: расширение Transact-SQL.
Полномочия	Блоки begin...end по умолчанию могут применять все пользователи. Для использования этих команд не требуется никаких полномочий.
См. также	**Команды** if...else

begin transaction

Описание

Отмечает начальную точку пользовательской транзакции.

Синтаксис

begin tran[saction] [*имя_транзакции*]

Параметры

имя_транзакции

Имя, присвоенное транзакции. Имя транзакции должно удовлетворять правилам для идентификаторов. Имена транзакций можно указывать только в самых внешних парах вложенных операторов begin transaction/commit или begin transaction/rollback.

Примеры

Следующий пример явно начинает транзакцию для оператора insert:

```
begin transaction
    insert into publishers (pub_id) values ("9999")
commit transaction
```

Использование

- Чтобы определить транзакцию, нужно заключить операторы SQL и/или системные процедуры между операторами begin transaction и commit. Если включен режим связанных транзакций, то СУБД Adaptive Server неявно вызывает begin transaction перед операторами delete, insert, open, fetch, select и update. Однако необходимо явно закончить транзакцию командой commit.
- Для отмены всей транзакции или ее части используется команда rollback. Команда rollback должна быть запущена из транзакции, нельзя выполнить откат транзакции после того, как она была зафиксирована.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду begin transaction по умолчанию могут выполнять все пользователи. Для ее использования не требуется каких-либо полномочий.

См. также

Команды commit, rollback, save transaction

break

Описание

Выполняет выход из цикла while. Команда break часто активируется по условию if.

Синтаксис

while *логическое_выражение*
оператор
break
оператор
continue

Параметры

логическое_выражение

Выражение (имя столбца, константа или любая комбинация имен столбцов и констант, связанных арифметическими или поразрядными операциями, либо подзапрос), возвращающее значения TRUE, FALSE или NULL. Если логическое выражение содержит оператор select, то он должен быть заключен в скобки.

Примеры

Если средняя цена меньше $30, то цены удваиваются. Затем выбирается максимальная цена. Если она меньше или равна $50, то цикл while перезапускается и цены еще раз удваиваются. Если максимальная цена превышает $50, то происходит выход из цикла while и выводится сообщение:

```
while (select avg(price) from titles) < $30
 begin
    update titles
    set price = price * 2
    select max(price) from titles
  if (select max(price) from titles) > $50
       break
    else
       continue
 end
 begin
  print "Too much for the market to bear"
 end
```

Использование

- Команда break приводит к выходу из цикла while. Затем выполняются операторы, которые стоят после ключевого слова end, помечающего конец цикла.
- Если два и более циклов while вложены друг в друга, то внутренняя команда break возвращает управление внешнему циклу, в который непосредственно вложен данный цикл.. Сначала будут выполнены все операторы, стоящие после окончания внутреннего цикла, затем этот внешний цикл будет перезапущен.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду break по умолчанию могут выполнять все пользователи. Для ее использования не требуется каких-либо полномочий.

См. также

Команды continue, while

case

Описание

Поддерживает условные выражения SQL; может использоваться везде, где допустимо использование значений выражения.

Синтаксис

```
case
        when условие_поиска then выражение
        [when условие_поиска then выражение]...
        [else выражение]
end
```

Синтаксис со значениями:

```
case выражение
        when выражение then выражение
        [ when выражение then выражение]...
        [else выражение]
end
```

Параметры

case
Начинает выражение case.

when
Предшествует условию поиска или выражению, с которым должно производиться сравнение.

условие_поиска
Используется для указания условия выбора результатов. Условия поиска для выражений case аналогичны условиям поиска в инструкции where. Условия поиска подробно рассматриваются в книге *Transact-SQL User's Guide*.

then
Предшествует выражению, которое указывает итоговое значение команды case.

выражение
Имя столбца, константа, функция, подзапрос или любая комбинация имен столбцов, констант и функций, связанных арифметическими или поразрядными операциями. Дополнительную информацию о выражениях см. в разделе "Выражения" на стр. 231.

Примеры

Пример 1. Выбор всех авторов из таблицы authors. Для определенных авторов указывается город, где они живут:

```
select au_lname, postalcode,
     case
            when postalcode = "94705"
                then "Berkeley Author"
            when postalcode = "94609"
```

```
                then "Oakland Author"
            when postalcode = "94612"
                then "Oakland Author"
            when postalcode = "97330"
                then "Corvallis Author"
        end
from authors
```

Пример 2. Запрос возвращает первое вхождение значения, не равного NULL, в столбце lowqty или highqty таблицы discounts:

```
select stor_id, discount,
        coalesce (lowqty, highqty)
from discounts
```

Пример 3. Альтернативный способ записи примера 2:

```
select stor_id, discount,
        case
            when lowqty is not NULL then lowqty
            else highqty
        end
from discounts
```

Пример 4. Выбор столбцов *title* (название) и *type* (тип) из таблицы *titles*. Если столбец *type* равен UNDECIDED, то nullif возвращает значение NULL:

```
select title,
        nullif(type, "UNDECIDED")
from titles
```

Пример 5. Альтернативный способ записи примера 4:

```
select title,
        case
            when type = "UNDECIDED" then NULL
            else type
        end
from titles
```

Использование

- Выражение case упрощает стандартные выражения SQL, поскольку позволяет представить условие поиска с помощью конструкции when...then вместо оператора if.
- Выражения case можно использовать везде, где в SQL разрешено использовать выражения.
- Хотя бы одно выражение должно быть отлично от ключевого слова NULL. Этот пример выдает следующее сообщение об ошибке:

```
select price, coalesce (NULL, NULL, NULL)
from titles

All result expressions in a CASE expression must not be NULL.
```

- Если запрос выдает несколько разных типов данных, то тип данных результата выражения case определяется иерархией типов данных, которая описана в разделе "Тип данных смешанных выражений" главы 1, "Системные и пользовательские типы данных" книги *Том 1, Основные конструкции*. Если указаны два типа данных, между которыми СУБД Adaptive Server не проводит неявных преобразований (например, *char* и *int*), то запрос не будет выполнен.
- coalesce – это сокращенная форма выражения case. В примере 3 приведена конструкция, альтернативная оператору coalesce.
- За оператором coalesce должно следовать не менее двух выражений. Этот пример выдает следующее сообщение об ошибке:

```
select stor_id, discount, coalesce (highqty)
from discounts

A single coalesce element is illegal in a COALESCE expression.
```

- nullif – это сокращенная форма выражения case.В примере 5 приведена конструкция, альтернативная оператору nullif.

Стандарты Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия Команду case по умолчанию могут выполнять все пользователи. Для ее использования не требуется каких-либо полномочий.

См. также **Команды** coalesce, nullif, if...else, select, where

checkpoint

Описание

Записывает все так называемые “грязные” страницы (которые были обновлены с момента их последней записи) на устройство хранения базы данных.

Синтаксис

контрольная точка

Примеры

Запись всех “грязных” страниц в текущей базе данных на устройство хранения базы данных независимо от системного графика контрольных точек:

```
контрольная точка
```

Использование

- Команда checkpoint используется только в качестве меры предосторожности в особых обстоятельствах. Например, Adaptive Server указывает, что следует выполнить команду checkpoint после изменения параметров базы данных.
- Команду checkpoint следует выполнять при каждом изменении параметров базы данных с помощью системной процедуры sp_dboption.

Автоматические контрольные точки

- Контрольные точки, выполняемые командой checkpoint, дополняют автоматические контрольные точки, которые выполняются с интервалами, вычисляемыми Adaptive Server на основе настраиваемого значения максимального допустимого времени восстановления.
- Команда checkpoint сокращает время автоматического восстановления за счет того, что в некоторой точке все завершенные транзакции были гарантированно записаны на устройство хранения базы данных. Типичная команда checkpoint выполняется примерно за 1 секунду, хотя время выполнения контрольных точек зависит от нагрузки на Adaptive Server.
- Интервал между выполнением автоматических контрольных точек вычисляется исходя из действий системы и значения интервала восстановления, хранящегося в системной таблице syscurconfigs. Интервал восстановления определяет частоту выполнения команды checkpoint, устанавливая максимальный период времени, который должен потребоваться для восстановления системы. Это значение можно изменить с помощью системной процедуры sp_configure.
- Если задаче housekeeper удается выгрузить все активные пулы буферов во всех настроенных кэшах во время бездействия сервера, то задача контрольной точки пробуждается. Она определяет, возможно ли записать контрольную точку базы данных.

Контрольные точки, созданные в результате выполнения задачи housekeeper, называются *произвольными контрольными точками*. Их выполнение не связано с записью большого количества “грязных” страниц на устройство хранения базы данных, поскольку задача housekeeper уже выполнила эту работу. Произвольные контрольные точки могут ускорить восстановление базы данных.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду checkpoint по умолчанию может выполнять только владелец базы данных. Это полномочие нельзя передать другим пользователям.

См. также

Системные процедуры sp_configure, sp_dboption

close

Описание	Закрывает курсор.
Синтаксис	close *имя_курсора*
Параметры	*имя_курсора* Имя курсора, который нужно закрыть.
Примеры	Закрытие курсора authors_crsr: `close authors_crsr`
Использование	• Команда close фактически удаляет результирующий набор курсора. Для закрытого курсора позиция курсора в наборе результатов становится неопределенной. • Если попытаться закрыть уже закрытый или несуществующий курсор, будет выдано сообщение об ошибке.
Стандарты	Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.
Полномочия	Команду close по умолчанию могут выполнять все пользователи. Для ее использования не требуется каких-либо полномочий.
См. также	**Команды** deallocate cursor, declare cursor, fetch, open

coalesce

Описание

Поддерживает условные выражения SQL; может использоваться в любом месте, где можно подставить значение выражения; альтернатива выражению case.

Синтаксис

coalesce (*выражение*, *выражение* [, *выражение*]...)

Параметры

coalesce

Вычисляет перечисленные выражения и возвращает первое значение, не равное NULL. Если все выражения равны NULL, то coalesce возвращает NULL.

выражение

Имя столбца, константа, функция, подзапрос или любая комбинация имен столбцов, констант и функций, связанных арифметическими или поразрядными операциями. Дополнительную информацию о выражениях см. в разделе “Выражения” на стр. 231.

Примеры

Пример 1. Запрос возвращает первое вхождение значения, не равного NULL, в столбце lowqty или highqty таблицы discounts:

```
select stor_id, discount,
        coalesce (lowqty, highqty)
from discounts
```

Пример 2. Альтернативный способ записи примера 1:

```
select stor_id, discount,
        case
            when lowqty is not NULL then lowqty
            else highqty
        end
from discounts
```

Использование

- Выражение coalesce упрощает стандартные выражения SQL, позволяя представить условие поиска как простое сравнение вместо применения конструкции when...then.
- Выражения coalesce можно использовать везде, где в SQL разрешено использовать выражения.
- Хотя бы один результат выражения coalesce должен вернуть значение, не равное NULL. Этот пример выдает следующее сообщение об ошибке:

```
select price, coalesce (NULL, NULL, NULL)
from titles

All result expressions in a CASE expression must not be NULL.
```

- Если запрос выдает несколько разных типов данных, то тип данных результата выражения case определяется иерархией типов данных, которая описана в разделе "Тип данных смешанных выражений" книги *Том 1, Основные конструкции*. Если указаны два типа данных, между которыми Adaptive Server не проводит неявных преобразований (например, *char* и *int*), то запрос не будет выполнен.
- coalesce – это сокращенная форма выражения case. В примере 2 приведена конструкция, альтернативная оператору coalesce.
- За оператором coalesce должно следовать не менее двух выражений. Этот пример выдает следующее сообщение об ошибке:

```
select stor_id, discount, coalesce (highqty)
from discounts

A single coalesce element is illegal in a COALESCE expression.
```

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду coalesce по умолчанию могут выполнять все пользователи. Для ее использования не требуется каких-либо полномочий.

См. также

Команды case, nullif, select, if...else, where

commit

Описание

Отмечает конечную точку пользовательской транзакции.

Синтаксис

commit [tran | transaction | work] [*имя_транзакции*]

Параметры

tran | transaction | work

Указывает, что транзакция или работа должна быть зафиксированы. Если указан tran, transaction или work, то можно также указать *имя_транзакции*.

имя_транзакции

Имя, присвоенное транзакции. Оно должно соответствовать правилам для идентификаторов. Имена транзакций используются только в самых внешних парах вложенных операторов begin transaction/commit или begin transaction/rollback.

Примеры

Обновление значений столбца royaltyper для двух авторов, вставка точки сохранения percentchanged. Затем определяется, каким образом 10-процентное повышение цены на книгу повлияет на размер авторских гонораров. После чего эта транзакция откатывается к точке сохранения по команде rollback transaction:

```
begin transaction royalty_change

update titleauthor
set royaltyper = 65
from titleauthor, titles
where royaltyper = 75
and titleauthor.title_id = titles.title_id
and title = "The Gourmet Microwave"

update titleauthor
set royaltyper = 35
from titleauthor, titles
where royaltyper = 25
and titleauthor.title_id = titles.title_id
and title = "The Gourmet Microwave"

save transaction percentchanged

update titles
set price = price * 1.1
where title = "The Gourmet Microwave"

select (price * total_sales) * royaltyper
from titles, titleauthor
```

```
where title = "The Gourmet Microwave"
and titles.title_id = titleauthor.title_id

rollback transaction percentchanged

commit transaction
```

Использование

- Чтобы определить транзакцию, нужно заключить между операторами begin transaction и commit операторы SQL и/или системные процедуры. Если включен режим связанных транзакций, то СУБД Adaptive Server неявно вызывает оператор begin transaction перед операторами delete, insert, open, fetch, select и update. Однако транзакция должна быть явно закончена командой commit.
- Для отмены всей транзакции или ее части используется команда rollback. Команда rollback должна быть запущена из транзакции. Нельзя выполнить откат транзакции после ввода команды commit.
- Если в данный момент активных транзакций нет, то команда commit или оператор rollback не влияют на работу СУБД Adaptive Server.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Формы commit transaction и commit tran этого оператора являются расширением Transact-SQL.

Полномочия

Команду commit по умолчанию могут выполнять все пользователи.

См. также

Команды begin transaction, rollback, save transaction

compute

Описание

Формирует итоговые значения, которые выводятся как дополнительные строки в результатах запроса.

Синтаксис

начало_оператора_select
compute *строка_агрегат* (*имя_столбца*)
[, *строка_агрегат*(*имя_столбца*)]...
[by *имя_столбца* [, *имя_столбца*]...]

Параметры

строка_агрегат

Одна из следующих функций:

Функция	Значение
sum	Сумма значений в (числовом) столбце
avg	Среднее значений в (числовом) столбце
min	Наименьшее значение в столбце
max	Наибольшее значение в столбце
count	Количество значений в столбце

имя_столбца

Имя столбца. Оно должно быть заключено в круглые скобки. Функции sum и avg можно использовать только для столбцов с числовыми значениями.

Одна инструкция compute может содержать несколько агрегатных функций, вычисляемых по одному и тому же набору столбцов (см. примеры 2 и 3). Чтобы создать несколько групп, нужно использовать несколько инструкций compute (см. пример 5).

by

Задает подгруппы, по которым вычисляются агрегатные значения. Для каждого набора строк, для которых значения в столбцах, указанных после ключевого слова by, одинаковы, генерируется новая строка с агрегатным значением. Если используется ключевое слово by, то необходимо также указать инструкцию order by.

Если после ключевого слова by перечислены несколько элементов, то группа разбивается на подгруппы и функция применяется на каждом уровне группировки.

Примеры

Пример 1. Вычисление суммы цен книг, которые стоят больше $12 для каждого типа кулинарных книг (то есть книг, для которых значение в столбце type заканчивается на “cook”):

```
select type, price
from titles
```

```
where price > $12
    and type like "%cook"
    order by type, price
compute sum(price) by type

type       price
---------  ------------
mod_cook          19.99
           sum
           ------------
                 19.99
type       price
---------  ------------
trad_cook         14.99
trad_cook         20.95
           sum
           ------------
                 35.94
(5 rows affected)
```

Пример 2. Вычисление суммы цен и авансов для книг, которые стоят больше $12, для каждого типа кулинарных книг:

```
select type, price, advance
from titles
where price > $12
    and type like "%cook"
    order by type, price
compute sum(price), sum(advance) by type

type       price      advance
---------  ---------  ------------
mod_cook       19.99          0.00
          sum         sum
           ---------  ------------
               19.99          0.00

type       price      advance
---------  ---------  ------------
trad_cook      14.99      8,000.00
trad_cook      20.95      7,000.00
           sum        sum
           ---------  ------------
               35.94     15,000.00
(5 rows affected)
```

Пример 3. Вычисление суммы цен и максимального аванса для книг, которые стоят больше $12, для каждого типа кулинарных книг:

```
select type, price, advance
from titles
where price > $12
    and type like "%cook"
    order by type, price
compute sum(price), max(advance) by type
```

```
type      price     advance
--------- --------- -------------
mod_cook      19.99         0.00
          sum
          ---------
              19.99
                    max
                    -------------
                             0.00

type      price     advance
--------- --------- -------------
trad_cook     14.99      8,000.00
trad_cook     20.95      7,000.00
          sum
          ---------
              35.94
                     max
                     -------------
                          8,000.00
(5 rows affected)
```

Пример 4. Разбивка данных на подгруппы по значениям столбцов type и pub_id и вычисление суммы цен на книги по психологии (то есть книги, для которых столбец type равен “psychology”) для каждой подгруппы:

```
select type, pub_id, price
from titles
where price > $10
    and type = "psychology"
    order by type, pub_id, price
compute  sum(price) by type, pub_id
```

```
type         pub_id    price
------------ --------- -----------
psychology   0736          10.95
psychology   0736          19.99
                         sum
                         ---------
```

```
                                30.94

type         pub_id    price
------------ --------- ---------
psychology   0877          21.59
                         sum
                         ---------
                             21.59
(5 rows affected)
```

Пример 5. Вычисление суммы цен всех книг по психологии, которые стоят больше $10, а также сумм в подгруппах с одинаковыми значениями столбцов type и pub_id:

```
select type, pub_id, price
from titles
where price > $10
    and type = "psychology"
order by type, pub_id, price
compute  sum(price) by type, pub_id
compute  sum(price) by type

type         pub_id    price
------------ --------- ---------
psychology   0736          10.95
psychology   0736          19.99
                         sum
                         ---------
                             30.94

type         pub_id    price
------------ --------- ---------
psychology   0877          21.59
                         sum
                         ---------
                             21.59
                         sum
                         ---------
                             52.53
(6 rows affected)
```

Пример 6. Вычисление суммы цен и суммы авансов всех кулинарных книг, которые стоят больше $10:

```
select type, price, advance
from titles
where price > $10
    and type like "%cook"
compute sum(price), sum(advance)
```

```
type       price        advance
---------  -----------  --------------
mod_cook         19.99            0.00
trad_cook        20.95        8,000.00
trad_cook        11.95        4,000.00
trad_cook        14.99        7,000.00
            sum          sum
           -----------  --------------
                 67.88       19,000.00
(5 rows affected)
```

Пример 7. Вычисление суммы цен на кулинарные книги и сумм выражений, включающих эти цены:

```
select type, price, price*2
from titles
    where type like "%cook"
compute sum(price), sum(price*2)
```

```
type         price
------------ --------------  ------------
mod_cook              19.99         39.98
mod_cook               2.99          5.98
trad_cook             20.95         41.90
trad_cook             11.95         23.90
trad_cook             14.99         29.98
             sum              sum
             ==============   ============
                     70.87        141.74
```

Использование

- Инструкция compute позволяет просмотреть строки данных и итоговые строки в одном наборе результатов. При помощи этой инструкции можно вычислять итоговые значения для нескольких подгрупп и несколько агрегатов для одной группы.
- Инструкцию compute можно использовать без ключевого слова by для вычисления общих итогов. В этом случае инструкция order by необязательна. См. пример 6.
- Если указана инструкция compute by, то необходимо также указать инструкцию order by. После инструкции compute by должны быть перечислены те же столбцы, что и после инструкции order by, или их подмножество. При этом столбцы в инструкции compute by должны располагаться в том же порядке слева направо, начинаться с того же выражения, ни одно выражение не должно быть пропущено. Например, если инструкция order by имеет вид `order by a, b, c`, то допустимо использовать следующие инструкции compute by:

```
compute by a, b, c
compute by a, b
compute by a
```

Ограничения

- В инструкции compute нельзя указывать более 127 столбцов для вычисления агрегатных функций.
- Инструкцию compute нельзя использовать в объявлении курсора.
- Можно вычислять итоговые значения и для выражений, и для столбцов. Все выражения и столбцы, указанные в инструкции compute, должны быть указаны и в списке выборки.
- Не допускается указывать псевдонимы столбцов в качестве аргументов агрегатных функций в инструкции compute, хотя псевдонимы можно указывать в списке выборки select, в инструкции order by и в инструкции by, являющейся частью инструкции compute.
- Нельзя указывать инструкцию select into в том же операторе, что и инструкцию compute, поскольку операторы, которые содержат compute, не создают обычные таблицы.
- Если инструкция compute содержит инструкцию group by, то должны быть выполнены следующие условия:
  - инструкция compute может содержать не более 255 агрегатов;
  - инструкция group by должна содержать не более 255 столбцов.
- Столбцы, включенные в инструкцию compute, не должны быть длиннее 255 байтов.

Результаты выполнения инструкции *compute* выдаются в виде новой строки или строк.

- Агрегатные функции обычно вычисляют одно значение для всех выбранных строк таблицы или для каждой группы, и эти итоговые значения отображаются как новые столбцы. Например:

```
select type, sum(price), sum(advance)
from titles
where type like "%cook"
group by type
type
------------  ---------  ----------
mod_cook           22.98  15,000.00
trad_cook          47.89  19,000.00

(2 rows affected)
```

- Инструкция compute позволяет извлекать строки данных и формировать итоговые строки в одной команде. Например:

```
select type, price, advance
from titles
where type like "%cook"
order by type
compute sum(price), sum(advance) by type
type         price       advance
----------   ----------  ----------------
mod_cook          2.99         15,000.00
mod_cook         19.99              0.00

Compute Result:
---------------------- -----------------
                 22.98         15,000.00
type          price          advance
----------   ----------  ----------------
trad_cook        11.95          4,000.00
trad_cook        14.99          8,000.00
trad_cook        20.95          7,000.00

Compute Result:
---------------------- -----------------
                 47.89         19,000.00
(7 rows affected)
```

- В таблице 7-4 описана группировка различных типов инструкций compute и формат выводимых ею данных.

Таблица 7-4. Инструкции compute by и строки данных

Инструкции и группировка	Формат выводимых данных	Примеры
Одна инструкция compute, одна и та же функция	Одна строка данных	1, 2, 4, 6, 7
Одна инструкция compute, разные функции	Одна строка данных на каждый тип функции	3
Несколько инструкций compute, одинаковые группируемые столбцы	Одна строка данных на каждую инструкцию compute; строки данных в выводе собраны вместе	Тот же результат, что и при использовании одной инструкции compute с разными функциями
Несколько инструкций compute, разные группируемые столбцы	Одна строка данных на каждую инструкцию compute; строки данных расположены в разных местах, в зависимости от группировки	5

Чувствительность к регистру

- Если на сервере используется порядок сортировки без учета регистра, то инструкция compute не учитывает регистр данных в указанных столбцах. Например, получив следующие данные:

```
select * from groupdemo
lname      amount
---------- ------------------
Smith                  10.00
smith                   5.00
SMITH                   7.00
Levi                    9.00
Lйvi                   20.00
```

инструкция compute by по столбцу lname выдаст следующий результат:

```
 select lname, amount from groupdemo
 order by lname
 compute sum(amount) by lname
 lname      amount
 ---------- -----------------------
 Levi                         9.00

 Compute Result:
 -----------------------
                      9.00

 lname      amount
 ---------- -----------------------
 Lйvi                        20.00

Compute Result:
 -----------------------
                     20.00

 lname      amount
 ---------- -----------------------
 smith                        5.00
 SMITH                        7.00
 Smith                       10.00

Compute Result:
 -----------------------
                     22.00
```

Тот же запрос на сервере, не учитывающем ригистр символов и диакритические знаки, дает следующий результат:

```
lname      amount
---------- ------------------------
Levi                         9.00
Lйvi                        20.00

Compute Result:
------------------------
                    29.00

lname      amount
---------- ------------------------
smith                        5.00
SMITH                        7.00
Smith                       10.00

Compute Result:
------------------------
                    22.00
```

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

См. также

Команды group by и having, select

Функции avg, count, max, min, sum

connect to...disconnect

Описание

Только для служб Component Integration Services – соединяет с указанным сервером и отключает подключенный сервер.

Синтаксис

connect to *имя_сервера*
disconnect

Параметры

имя_сервера
Сервер, с которым требуется установить транзитное соединение.

Примеры

Пример 1. Установка транзитного соединения с сервером с именем SYBASE:

```
connect to SYBASE
```

Пример 2. Отключение подключенного сервера:

```
disconnect
```

Использование

- Команда connect to указывает сервер, с которым требуется установить транзитное соединение. Режим транзитной пересылки позволяет выполнять собственные операции на удаленном сервере.
- *имя_сервера* должно быть одним из имен серверов в системной таблице sysservers, для которых определены сетевое имя и класс сервера.
- При подключении к серверу *имя_сервера* от имени пользователя службы Component Integration Services используют один из следующих идентификаторов:
  - псевдоним для удаленного входа в систему, описанный в таблице sysattributes, если таковой имеется;
  - имя пользователя и пароль.

  В любом случае, если соединение с указанным сервером не может быть установлено, возвращается сообщение об ошибке.
- Дополнительную информацию о добавлении удаленных серверов см. в описании процедуры sp_addserver.
- После того как транзитное соединение установлено, службы Component Integration Services не обрабатывают получаемый в дальнейшем языковой текст средствами синтаксического анализатора и компилятора Transact-SQL. Они передают операторы прямо на указанный сервер и преобразуют результаты в форму, которую может распознать интерфейс Open Client и возвратить в клиентскую программу.

- Для отключения соединения, созданного командой connect to, используется команда disconnect. Эту команду можно использовать, только если соединение было установлено командой connect to.
- Можно использовать сокращенную форму команды disconnect – disc.
- Команда disconnect возвращает сообщение об ошибке, если ранее не была дана команда connect to и сервер не был подключен к какому-либо удаленному серверу.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Полномочия на использование команды connect to должен явно предоставлять системный администратор. При этом используется следующий синтаксис:

grant connect to *имя_пользователя*

Системный администратор может глобально предоставлять или отзывать полномочия на подключение группе public, когда он находится в базе данных master. Системный администратор может предоставить или отозвать полномочия на использование команды connect to пользователю, зарегистрированному в базе данных master. Для этого системный администратор сначала должен отозвать полномочия у группы public следующим образом:

```
use master
go
revoke connect from public
go
sp_adduser fred
go
grant connect to fred
go
```

См. также

Команды create existing table, grant

Системные процедуры sp_addserver, sp_autoconnect, sp_helpserver, sp_passthru, sp_remotesql, sp_serveroption

continue

Описание

Перезапускает цикл while. Команда continue часто активируется по условию if.

Синтаксис

```
while логическое_выражение
        оператор
    break
        оператор
continue
```

Примеры

Если средняя цена меньше $30, то цены удваиваются. Затем выбирается максимальная цена. Если она меньше или равна $50, то цикл while перезапускается и цены удваиваются еще раз. Если максимальная цена превышает $50, то происходит выход из цикла while с сообщением:

```
while (select avg(price) from titles) < $30
begin
  update titles
  set price = price * 2
  select max(price) from titles

  if (select max(price) from titles) > $50
     break
  else
     continue
end

begin
print "Too much for the market to bear"
end
```

Использование

- Команда continue перезапускает цикл while, пропуская любые операторы после continue.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду continue по умолчанию могут выполнять все пользователи. Для ее использования не требуется каких-либо полномочий.

См. также

Команды break, while

create database

Описание

Создает новую базу данных.

Синтаксис

```
create database имя_базы_данных
        [on {default | устройство_базы_данных} [= размер]
            [, устройство_базы_данных [= размер]]...]
        [log on устройство_базы_данных [= размер]
            [, устройство_базы_данных [= размер]]...]
        [with {override | default_location = "путь_ к_ файлу"}]
        [for {load | proxy_update}]
```

Параметры

имя_базы_данных

Имя новой базы данных. Должно быть допустимым идентификатором, не может быть переменной.

on

Определяет местоположение и размер базы данных.

default

Указывает, что оператор create database может разместить новую базу данных на любом устройстве по умолчанию для хранения базы данных (со значением default в столбце sysdevices.status). Чтобы указать размер базы данных, не указывая местоположения, используйте следующий синтаксис:

```
on default = размер
```

Чтобы изменить состояние устройства для хранения базы данных (столбец sysdevices.status) на default, нужно выполнить процедуру sp_diskdefault.

устройство_базы_данных

Логическое имя устройства, где будет размещаться база данных. База данных может занимать разный объем пространства на нескольких устройствах. Для добавления устройств базы данных к серверу Adaptive Server используйте команду disk init.

размер

Размер пространства, которое нужно выделить для расширения базы данных. Размер может быть указан в следующих единицах измерения: ‘k’ или ‘K’ (килобайты), ‘m’ или ‘M’ (мегабайты), ‘g’ или ‘G’ (гигабайты). Sybase рекомендует всегда указывать единицу измерения. Если единица измерения не указывается, кавычки использовать необязательно. В противном случае размер обязательно должен быть заключен в кавычки.

log on
Задает логическое имя устройства, где размещаются журналы базы данных. В инструкции log on можно указать несколько устройств.

with override
Заставляет Adaptive Server использовать указанную спецификацию устройств, даже если в соответствии с ней данные и журналы транзакций будут храниться на одном устройстве, что ставит под угрозу возможность восстановления базы данных на текущий момент времени. Если попытаться разместить данные и журналы на одном устройстве, не указав этой инструкции, команда create database не будет выполнена. Если же попытаться разместить данные и журналы на одном устройстве и указать инструкцию with override, то будет выдано соответствующее предупреждение, но команда будет выполнена успешно.

for load
Вызывает специальную версию команды create database, которая используется только для загрузки резервной копии базы данных. Дополнительную информацию см. в разделе “Использование параметра for load” на стр. 344.

with default_location
Определяет место хранения новых таблиц. Кроме того, если указать инструкцию for proxy_update, в указанном местоположении будет автоматически создано по одной прокси-таблице для каждой удаленной таблицы или представления.

for proxy_update
Автоматически получает метаданные из удаленного местоположения и создает прокси-таблицы. Инструкцию for proxy_update нельзя использовать, не указав инструкцию with default_location.

Примеры

Пример 1. Создание базы данных pubs:

```
create database pubs
```

Пример 2. Создание базы данных pubs размером 4 МБ:

```
create database pubs
on default = 4
```

Пример 3. Создание базы данных pubs, 3 МБ данных которой будут помещены в сегмент datadev и 2 МБ – в сегмент moredatadev:

```
create database pubs
on datadev = 3, moredatadev = 2
```

Пример 4. Создание базы данных pubs, 3МБ данных которой будут помещены в сегмент datadev, а 1 МБ журнала – в сегмент logdev:

```
create database pubs
on datadev = 3
log on logdev = 1
```

Пример 5. Создание прокси-БД proxydb без автоматического создания прокси-таблиц:

```
create database proxydb
with default_location
"UNITEST.pubs.dbo."
```

Пример 6. Создание прокси-БД proxydb с автоматическим созданием прокси-таблиц:

```
create database proxydb
on default = 4
with default_location
"UNITEST.pubs2.dbo."
for proxy_update
```

Использование

- Команду create database нужно выполнять из базы данных master.
- Можно задать *размер* как значение типа float, но оно будет округлено в меньшую сторону до ближайшего числа, кратного единице выделения пространства.
- Если размер базы данных не указан явно, то он определяется размером базы данных model. Минимальный размер равен четырем единицам выделения пространства.
- Так как Adaptive Server выделяет пространство под базы данных при выполнении команд create database и alter database участками по 256 логических страниц, эти команды округляют указанный размер в меньшую сторону до ближайшей величины, кратной единице выделения пространства.
- Если единица измерения не указана, Adaptive Server считает, что размер указан в мегабайтах, и преобразует его к размеру логических страниц, используемых сервером.
- Если местоположение и размер базы данных не указаны, местоположением по умолчанию считается любое устройство базы данных по умолчанию, содержащееся в таблице master..sysdevices. Размер по умолчанию равен большему из следующих значений: размер базы данных model и значение параметра default database size из таблицы sysconfigures.

Системный администратор может увеличить размер по умолчанию, изменив значение параметра default database size процедурой sp_configure и перезапустив Adaptive Server. Значение параметра default database size должно быть не меньше размера базы данных. Когда размер базы данных увеличивается, значение этого параметра также нужно увеличить.

Если Adaptive Server не может предоставить запрашиваемый объем пространства, он выделяет максимально возможный объем пространства на каждом устройстве и печатает сообщение о том, сколько пространства было выделено и где оно расположено. Максимальный размер базы данных зависит от системы.

- Если прокси-БД была создана командой

  ```
  create database mydb on my_device
  with default_location = "путь" for proxy_update
  ```

  вычисление размера базы данных не будет выполнено (поскольку указано имя устройства). В результате этого команда может не выполниться, если размер по умолчанию (размер) недостаточен, чтобы вместить все прокси-таблицы.

  Чтобы службы CIS смогли оценить размер базы данных, не нужно указывать имя устройства или другие параметры:

  ```
  create database mydb
  with default_location = "путь" for proxy_update
  ```

Ограничения

- Adaptive Server позволяет создавать до 32 767 баз данных.
- Adaptive Server не может создавать несколько баз данных одновременно. Если два запроса на создание базы данных приходят одновременно, один из пользователей получает сообщение:

  ```
  model database in use: cannot create new database
  ```

- Пространство, выделяемое на устройстве базы данных командой create database или alter database, представляет собой фрагмент устройства. Для каждого такого выделения пространства в таблицу sysusages записывается отдельная строка.
- Максимальное количество именованных сегментов для базы данных равно 32. Сегменты являются именованными подмножествами устройств базы данных, доступных определенному серверу Adaptive Server. Дополнительную информацию о сегментах см. в книге *Руководство по системному администрированию*.

Создание новых баз данных из базы данных *model*

- Новая база данных создается путем копирования базы данных model.
- Базу данных model можно настраивать, добавляя в нее таблицы, хранимые процедуры, пользовательские типы данных и другие объекты и изменяя значения ее параметров. Новые базы данных будут наследовать новые объекты и настройки базы данных model.
- Для обеспечения восстанавливаемости команда create database очищает все страницы, которые не были инициализированы при копировании базы данных model. Этот процесс может занять несколько минут, в зависимости от размера базы данных и производительности операционной системы.

  Если база данных создается для загрузки в нее резервной копии, укажите параметр for load, чтобы пропустить этап очистки страниц. Это заметно ускорит процесс создания базы данных.

Обеспечение восстанавливаемости базы данных

- При каждом создании базы данных следует сделать резервную копию базы данных master. Это упрощает восстановление базы данных master в случае ее повреждения и делает это восстановление более надежным.

  Примечание. Если вы создали новую базу данных, но не создали резервную копию базы данных master, то, возможно, изменения можно будет восстановить командой disk reinit.

- Инструкция with override позволяет разместить сегменты данных и журнала на одном устройстве. Однако чтобы обеспечить полную восстанавливаемость, в инструкции log on нужно указать устройство (устройства), на котором (которых) не хранятся данные. В случае поломки жесткого диска база данных может быть восстановлена из резервной копии и журналов транзакций.

  Для небольшой базы данных журналы и данные можно поместить на одном устройстве, но тогда *единственной* возможностью восстановления будут резервные копии, созданные с помощью команды dump database.

- Размер устройства, требующийся для хранения журнала транзакций зависит от объема операций обновления и от частоты создания резервных копий журнала транзакций. Как правило, под устройство для журналов выделяют 10 – 25% от размера самой базы данных. Первоначальный размер следует сделать небольшим, так как пространство, выделенное под журналы транзакций, не может быть освобождено и использовано впоследствии для хранения данных.

Использование параметра *for load*

Параметр for load позволяет восстановить базу данных после сбоя носителя или перенести базу данных с одного компьютера на другой, если к базе данных не был добавлен сегмент процедурой sp_addsegment. С помощью команды alter database for load можно создать новую базу данных по образу загружаемой резервной копии. При загрузке резервной копии в новую базу данных нужно заново повторить процедуру выделения пространства, сделанную для исходной базы данных. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

- Если база данных была создана с параметром for load, то до тех пор, пока в нее не будет загружена резервная копия, можно выполнять только следующие команды:

  - alter database for load
  - drop database
  - load database

  После загрузки резервной копии в базу данных можно выполнять некоторые разновидности команды dbcc. После выполнения команды online database ограничения на выполнение других команд снимаются.

- Для базы данных, при создании которой был указан параметр for load, столбец status в выходных данных системной процедуры sp_helpdb будет равен “don’t recover”.

Получение информации о базах данных

- Для получения отчета о базе данных выполните системную процедуру sp_helpdb.
- Для получения отчета о пространстве, занимаемом базой данных, выполните системную процедуру sp_spaceused.

Использование инструкций *with default_location* и *for proxy_update*

Если не указана инструкция for proxy_update, то инструкция with default_location дает тот же результат, что и системная процедура sp_defaultloc: для таблиц, создаваемых командами create existing table и create table, устанавливается местоположение по умолчанию, но при выполнении команды create database определения прокси-таблиц не импортируются автоматически.

- Если инструкция for proxy_update указывается без инструкции default_location, выдается сообщение об ошибке.

- Когда создается прокси-БД (с помощью параметра for proxy_update), вызываются службы Component Integration Services для выполнения следующих действий:
  - Оценки размера базы данных, требующегося для размещения всех прокси-таблиц, соответствующих таблицам и представлениям в базе данных главного сервера. Эта оценка равна количеству страниц базы данных, необходимых для размещения всех прокси-таблиц и индексов. Оценка используется, только если не указаны размер и устройства для хранения базы данных.
  - Создания всех прокси-таблиц, соответствующих фактическим таблицам и представлениям в базе данных на первичном сервере.
  - Предоставления всем пользователям всех полномочий для доступа к прокси-таблицам.
  - Добавления пользователя guest в прокси-БД.
  - Состояние базы данных изменится на 'Is_A_Proxy'. Это состояние хранится в столбце master.dbo.sysdatabases.status4.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду create database по умолчанию могут выполнять системные администраторы. Они могут передать эти полномочия другим пользователям, которые перечислены в таблице sysusers базы данных master. Как правило, полномочия на выполнение команды create database не назначаются другим пользователям, чтобы сохранить контроль над выделением пространства для базы данных.

Полномочием на создание базы данных sybsecurity обладает только администратор безопасности системы.

Полномочия на выполнение команды create database не предоставляются командой grant all.

См. также

Команды alter database, disk init, drop database, dump database, load database, online database

Системные процедуры sp_changedbowner, sp_diskdefault, sp_helpdb, sp_logdevice, sp_renamedb, sp_spaceused

create default

Описание

Определяет значение, которое будет вставлено в столбец (или во все столбцы пользовательского типа данных), если значение не указано явно при вставке строки в таблицу.

Синтаксис

create default [*владелец*.]*имя_значения_по_умолчанию*
as *константа*

Параметры

имя_значения_по_умолчанию

Имя значения по умолчанию. Должно быть допустимым идентификатором и не может быть переменной. Чтобы создать значение по умолчанию с тем же именем, что и существующее значение по умолчанию, принадлежащее другому пользователю в текущей базе данных, перед именем значения по умолчанию нужно указать имя владельца. По умолчанию *владелец* – текущий пользователь.

константа

Выражение, не содержащее имен столбцов и других объектов базы данных. Оно также не может содержать глобальные переменные, но может содержать встроенные функции, не ссылающиеся на объекты базы данных. Символьные выражения и даты следует заключать в кавычки, а для двоичных констант использовать префикс “0x”.

Примеры

Пример 1. Определение значения по умолчанию. После этого его можно назначить столбцу или пользовательскому типу данных с помощью системной процедуры sp_bindefault:

```
create default phonedflt as "UNKNOWN"

sp_bindefault phonedflt, "authors.phone"
```

Значение по умолчанию используется, когда значение для столбца phone таблицы authors не указывается явно при вводе данных. Отсутствие указанного значения не равнозначно вводу значения NULL. Чтобы занести в столбец заданное для него значение по умолчанию, нужно выполнить команду insert со списком столбцов, не включающих этот столбец.

Пример 2. Создание значения по умолчанию todays_date, вставляющего текущую дату в столбцы, которым оно назначено:

```
create default todays_date as getdate()
```

Использование

- Процедура sp_bindefault позволяет привязать значение по умолчанию к столбцу или пользовательскому типу данных (но не к встроенным типам данных Adaptive Server).
- К типу данных можно привязать новое значение по умолчанию, не отменяя привязки старого. При этом новое значение заменит старое.

- Системная процедура sp_hidetext позволяет скрыть исходный текст значения по умолчанию.

Ограничения

- Значение по умолчанию можно создать только в текущей базе данных.
- Команды create default нельзя объединять с другими командами в один пакет.
- Необходимо удалить старое значение по умолчанию командой drop default, прежде чем создать новое с тем же именем; перед удалением значения по умолчанию нужно отменить его привязки к столбцам и типам данных с помощью процедуры sp_unbindefault.

Совместимость типов данных

- При попытке вставить значение по умолчанию, не совместимое с типом данных столбца, Adaptive Server генерирует сообщение об ошибке. Если к столбцу типа integer было привязано символьное выражение, например “N/A”, то команда *insert* , в которой не указано значение этого столбца, не будет выполнена.
- Если длина значения по умолчанию больше длины символьного столбца, Adaptive Server отсекает часть строки либо генерирует исключение в зависимости от значения параметра *string_rtruncation*. См. также описание команды set.

Получение информации о значениях по умолчанию

- Описания значений по умолчанию хранятся в таблице syscomments.
- После того как значение по умолчанию было привязано к столбцу, его идентификатор сохраняется в таблице syscolumns. Если же привязка значения по умолчанию осуществляется к пользовательскому типу данных, его объектный идентификатор сохраняется в таблице systypes.
- Для переименования значения по умолчанию используется процедура sp_rename.
- Для получения информации о тексте значения по умолчанию используется процедура sp_helptext.

Значения по умолчанию и правила

- Значение по умолчанию не должно нарушать правило, если они оба привязаны к одному столбцу. Значение по умолчанию, нарушающее правило, не вставляется. При попытке вставить такое значение Adaptive Server генерирует сообщение об ошибке.

Значения по умолчанию и значения NULL

- Если для столбца, не допускающего значений NULL, не задано значение по умолчанию, то при попытке вставить строку, не указав значение для этого столбца, Adaptive Server сгенерирует сообщение об ошибке.

  Таблица 7-5 иллюстрирует отношения между существованием значений по умолчанию и определением столбца (NULL или NOT NULL).

Таблица 7-5. Отношение между допустимостью значений NULL в столбце и значениями столбцов по умолчанию

Допустимость значений NULL в столбце	Значение не указано, значение по умолчанию не задано	Значение не указано, значение по умолчанию задано	Вводится значение NULL, значение по умолчанию не задано	Вводится значение NULL, значение по умолчанию задано
NULL	Вставляется значение NULL	Вставляется значение по умолчанию	Вставляется значение NULL	Вставляется значение NULL
NOT NULL	Ошибка, команда не выполняется	Вставляется значение по умолчанию	Ошибка, команда не выполняется	Ошибка, команда не выполняется

Задание значения по умолчанию в операторе *create table*

- Значение по умолчанию можно определить не только с помощью команды create default, но и с помощью инструкции *default* команды create table. Однако такие значения по умолчанию относятся только к таблице, создаваемой командой create table, и их нельзя привязать к другим таблицам. Информацию об ограничениях целостности см. в описании команд create table и alter table.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Чтобы создать значения по умолчанию, совместимые с SQL92, нужно использовать инструкцию default оператора create table.

Полномочия

Команду create default по умолчанию может выполнять владелец базы данных. Он может передать эти полномочия другим пользователям.

Команды alter table, create rule, create table, drop default, drop rule

Системные процедуры sp_bindefault, sp_help, sp_helptext, sp_rename, sp_unbindefault

create existing table

Описание

Только для служб Component Integration Services Создает прокси-таблицу, извлекает и сохраняет метаданные из удаленной таблицы и помещает данные в прокси-таблицу. Позволяет привязать прокси-таблицу к таблице, представлению или процедуре, находящейся на удаленном узле.

Синтаксис

```
create existing table имя_таблицы (список_столбцов)
        [ on имя_сегмента ]
        [ [ external {table | procedure | file} ] at путь ]
```

Параметры

имя_таблицы
Имя таблицы, для которой создается прокси-таблица.

список_столбцов
Название списка столбцов, в которых хранится информация об удаленной таблице.

on имя_сегмента
Сегмент, содержащий удаленную таблицу.

external
Указывает, что объект является удаленным.

table
Указывает, что удаленный объект – таблица или представление. Значение по умолчанию – external table.

procedure
Указывает, что удаленный объект – хранимая процедура.

file
Указывает, что удаленный объект – файл.

at путь
Определяет местоположение удаленного объекта. Параметр *путь* имеет формат *имя_сервера.имя_базы_данных.владелец.объект*, где:

- *имя_сервера* (указывать обязательно) – имя сервера, содержащего удаленный объект;
- *имя_базы_данных* (указывать необязательно) – имя базы данных, управляемой удаленным сервером, содержащим удаленный объект;
- *владелец* (указывать необязательно) – имя пользователя удаленного сервера, владеющего удаленным объектом;
- *объект* (указывать обязательно) – имя удаленной таблицы, представления или процедуры.

Примеры

Пример 1. Создание прокси-таблицы authors:

```
create existing table authors
(
au_id        id,
au_lname     varchar(40)    NOT NULL,
au_fname     varchar(20)    NOT NULL,
phone        char(12),
address      varchar(40)    NULL,
city         varchar(20)    NULL,
state        char(2)        NULL,
zip          char(5)        NULL,
contract     bit
)
```

Пример 2. Создание прокси-таблицы syb_columns:

```
create existing table syb_columns
(
id          int,
number      smallint,
colid       tinyint,
status      tinyint,
type        tinyint,
length      tinyint,
offset      smallint,
usertype    smallint,
cdefault    int,
domain      int,
name        varchar(30),
printfmt    varchar(255)    NULL,
prec        tinyint         NULL,
scale       tinyint         NULL
)
```

Пример 3. Создание прокси-таблицы blurbs для таблицы blurbs, расположенной на удаленном сервере SERVER_A:

```
create existing table blurbs
(author_id               id        not null,
copy               text        not null)
at "SERVER_A.db1.joe.blurbs"
```

Пример 4. Создание прокси-таблицы rpc1 для удаленной процедуры p1:

```
create existing table rpc1
(column_1               int,
column_2               int)
external procedure
at "SERVER_A.db1.joe.p1"
```

Использование

- Оператор create existing table не создает новую таблицу, если удаленный объект не является файлом. Вместо этого службы Component Integration Services проверяют привязку таблицы (чтобы убедиться, что информация, содержащаяся в *списке_столбцов*, соответствует удаленной таблице), проверяют существование указанного объекта и извлекают и сохраняют метаданные об удаленной таблице.
- Если файл-источник или удаленный объект не существует, команда возвращает сообщение об ошибке и не выполняется.
- Если объект существует, обновляются системные таблицы sysobjects, syscolumns и sysindexes. Операция проверки состоит из следующих этапов:
  a Определение природы существующего объекта. Для файлов-источников определяется организация файла и формат записи. Для объектов на удаленном сервере определяется, чем является удаленный объект: таблицей, представлением или удаленной процедурой.
  b Для объектов на удаленном сервере, не являющихся удаленными процедурами, атрибуты столбцов удаленной таблицы или представления сравниваются с описаниями, указанными в параметре список_столбцов.
  c Из файла-источника или удаленной таблицы извлекается информация об индексах, которая необходима для создания строк в системной таблице sysindexes. Это позволяет определять индексы и ключи в терминах Adaptive Server, благодаря чему оптимизатор запросов может учитывать индексы, которые могли быть построены по этой таблице.
- Инструкция on имя_сегмента обрабатывается локально и не передается на удаленный сервер.
- После успешного определения существующей таблицы вызовите для нее команду update statistics. Это позволит оптимизатору запросов делать разумный выбор индексов и порядка соединения.
- Службы Component Integration Services позволяют создать прокси-таблицу, включающую столбец с определением NOT NULL, даже если удаленный столбец определен как NULL. В этом случае появится сообщение о несоответствии.
- Информация о расположении, указанная в ключевом слове at, совпадает с данными, предоставляемыми системной процедурой sp_addobjectdef. Эта информация хранится в таблице sysattributes.

- Службы Component Integration Services вставляют или обновляют запись в таблице systabstats для каждого индекса удаленной таблицы. Так как подробная структурная статистика неприменима к удаленным индексам, в таблице systabstats делаются записи только в минимальное количество столбцов: id, indid и rowcnt.
- Внешние файлы не могут иметь тип text, image или Java ADT.

Преобразование типов данных

- При выполнении команды create existing table необходимо использовать типы данных, распознаваемые Adaptive Server. Если удаленные таблицы расположены на сервере другого класса, то при извлечении данных происходит автоматическое преобразование типов данных удаленной таблицы в типы Adaptive Server. Если преобразование невозможно, службы Component Integration Services не позволяют определить таблицу.
- Описание всех поддерживаемых классов сервера и возможных преобразований типов данных, неявно выполняемых Component Integration Services, содержится в книге *Component Integration Service User's Guide*.

Изменения, зависящие от класса сервера

- Сервер любого класса позволяет указать меньше столбцов, чем содержится в таблице удаленного сервера.
- Сервер любого класса подбирает столбцы по именам.
- Сервер любого класса позволяет определять тип данных столбца как тип, допускающий двунаправленное преобразование в тип столбца удаленной таблицы.

Удаленные процедуры

- Если прокси-таблица – таблица процедурного типа, необходимо предоставить список столбцов, соответствующий описанию результирующего набора удаленной процедуры. Команда create existing table *не* проверяет правильность списка столбцов.
- Для процедур индексы не создаются.
- Службы Component Integration Services позволяют работать с результирующим набором удаленной процедуры как с виртуальной таблицей, в частности сортировать его, соединять с другими таблицами и вставлять в другую таблицу операторами insert или select. Однако таблица процедурного типа предназначена только для чтения, поэтому с ней нельзя выполнить следующие команды:
  - alter table

- create index
- delete
- insert
- truncate table
- update

- Имя столбца должно начинаться с символа "_" (подчеркивание), чтобы указать, что столбец не является частью результирующего набора удаленной процедуры. Эти столбцы используются в качестве столбцов параметров. Например:

```
create existing table rpc1
(
    a        int,
    b        int,
    c        int,
    _p1        int null,
    _p2        int null
)
external procedure
at "SYBASE.sybsystemprocs.dbo.myproc"
```

  В этом примере столбцы _p1 и _p2 являются входными параметрами. Они не входят в результирующий набор, но могут участвовать в запросе.

```
select a, b, c from t1
where _p1 = 10 and _p2 = 20
```

  Службы Component Integration Services передают аргументы поиска удаленной процедуре как параметры, используя имена *@p1* и *@p2*.

- Описания столбцов-параметров в операторе create existing table должны удовлетворять следующим правилам:
  - Столбец-параметр должен допускать значения NULL.
  - Столбцы-параметры не могут предшествовать обычным столбцам результата, они должны находится в конце списка столбцов.
- Если столбец-параметр включается в список оператора select *и* передается удаленной процедуре в качестве параметра, возвращаемое значение присваивается инструкцией where.
- Если столбец-параметр включен в список оператора select, но отсутствует в инструкции where или не может быть передан удаленной процедуре в качестве параметра, его значение – NULL.

- Столбец-параметр можно передать удаленной процедуре в качестве параметра, если процессор запросов Adaptive Server считает его аргументом, подходящим для поиска информации. Столбец-параметр считается подходящим для поиска аргументом, если он не включен ни в один предикат or. Например, предикат or во второй строке следующего запроса не позволяет использовать столбцы-параметры в качестве параметров:

```
select a, b, c from t1
where _p1 = 10 or _p2 = 20
```

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на выполнение команды create existing table принадлежат владельцу таблицы и не могут быть переданы другим пользователям.

См. также

Команды alter table, create table, create proxy_table, drop index, insert, order by, set, update

create function (SQLJ)

Описание

Создает пользовательскую функцию путем добавления SQL-оболочки к статическому методу Java. Может возвращать значение, определенное методом.

Синтаксис

```
create function [владелец.]имя_функции_sql
        ( [имя_параметра_sql тип_данных_sql
            [( длина)| (точность[, масштаб]) ]
        [ [, имя_параметра_sql тип_данных_sql
            [( длина)| (точность[, масштаб]) ]]
        ... ] ] )
    returns тип_данных_sql
        [ ( длина)| (точность[, масштаб]) ]
    [modifies sql data]
    [returns null on null input |
        called on null input]
    [deterministic | not deterministic]
    [exportable]
    language java
    parameter style java
    external name 'имя_метода_java
        [ ( [тип_данных_java[, тип_данных_java
        ...] ] ) ] '
```

Параметры

имя_функции_sql

Имя функции Transact-SQL. Должно быть допустимым идентификатором, не может быть переменной.

имя_параметра_sql

Имя аргумента функции. Значения входных параметров указываются при вызове функции. Параметры указывать необязательно: функция SQLJ может не иметь аргументов.

Имена параметров должны соответствовать правилам для идентификаторов. Если значение параметра содержит не буквенно-цифровые символы, оно должно быть заключено в кавычки. Это правило распространяется на имена объектов, дополняемые именем базы данных или владельца, так как они содержат точку. Значение параметра, начинающееся с цифры, также должно быть заключено в кавычки.

тип_данных_sql [(*длина*) | (*точность* [, *масштаб*])]

Тип данных Transact-SQL для параметра. Подробное описание этих параметров см. в разделе create procedure на стр. 375.

тип_данных_sql является сигнатурой процедуры SQL.

returns тип_данных_sql

Определяет тип результата функции.

modifies sql data
: Указывает, что метод Java вызывает операции SQL, читает и изменяет данные в базе данных. Это единственно возможная настройка, которая также и является настройкой по умолчанию. Она обеспечивает синтаксическую совместимость со стандартом ANSI.

deterministic | not deterministic
: Эти ключевые слова используются для синтаксической совместимости со стандартом ANSI. К настоящему моменту задаваемые ими действия не реализованы.

exportable
: Указывает, что процедуру необходимо запустить на удаленном сервере с использованием Adaptive Server OmniConnect™. Процедура и метод, на котором она построена, должны располагаться на удаленном сервере.

language java
: Указывает, что внешняя подпрограмма написана на языке Java. Эта инструкция является обязательной для функций SQLJ.

parameter style java
: Указывает, что параметры, передаваемые внешней подпрограмме во время выполнения, являются параметрами Java. Эта инструкция является обязательной для функций SQLJ.

external
: Указывает, что команда create function определяет SQL-имя для внешней подпрограммы, написанной на языке, отличном от SQL.

name
: Определяет имя внешней подпрограммы (метода Java). Имя указывается в формате ‘имя_метода_java [тип_данных_java[{, тип_данных_java} ...]]’. Оно является символьным литералом, заключенным в одинарные кавычки.

имя_метода_java
: Определяет имя внешнего метода Java.

тип_данных_java
: Указывает тип данных Java, совместимый в обычном смысле (mappable) или совместимый по набору результатов (result-set mappable). Является сигнатурой метода Java.

Примеры

Создание функции square_root, вызывающей метод java.lang.Math.sqrt():

```
create function square_root
    (input_number double precision) returns
       double precision
    language java parameter style java
    external name 'java.lang.Math.sqrt'
```

Использование

- Имя функции SQLJ не должно совпадать с именем какой-либо встроенной функции Adaptive Server.
- Можно создавать пользовательские функции (основанные на статических методах Java) и функции SQLJ с одними и теми же именами классов и методов.

  Примечание. Порядок поиска, принятый в Adaptive Server, гарантирует, что первой будет найдена функция SQLJ.

- Вы можете указать до 31 параметра в операторе create function.
- Подробности о команде create function см. в книге *Java in Adaptive Server Enterprise*.

Полномочия

Только владелец базы данных и пользователь с ролью sa могут выполнять команду create function. Владелец базы данных и пользователь с ролью sa не могут передавать полномочия на выполнение команды create function другим пользователям.

См. также

Команды create function (SQLJ), drop function (SQLJ)

Системные процедуры sp_depends, sp_help, sp_helpjava, sp_helprotect

create index

Описание

Создает индекс по одному или нескольким столбцам таблицы.

Синтаксис

```
create [unique] [clustered | nonclustered]
        index имя_индекса
    on [[база_данных.]владелец.]имя_таблицы
        (имя_столбца [asc | desc]
            [, имя_столбца [asc | desc]]...)
    [with { fillfactor = pct,
            max_rows_per_page = количество_строк,
            reservepagegap = количество_страниц,
            consumers = x, ignore_dup_key, sorted_data,
            [ignore_dup_row | allow_dup_row],
             statistics using количество_шагов values  } ]
    [on имя_сегмента ]
```

Параметры

unique

Запрещает одинаковые значения в индексе (называемые также “значениями ключа”). Наличие одинаковых значений ключа проверяется при создании индекса (если данные уже существуют) и при добавлении данных операторами insert и update. Если в нескольких строках значения ключа одинаковы или несколько строк содержат значения NULL, команда не выполняется и возвращается сообщение об ошибке с указанием повторяющегося значения.

Предупреждение. Одинаковые строки не будут обнаружены, если таблица содержит столбцы типа text или image со значениями, отличными от NULL.

Команды update и insert, в результате выполнения которых в таблице будут строки с одинаковыми значениями ключа, вызовут ошибку, если индекс был создан без ключевых слов ignore_dup_row и ignore_dup_key.

Составные индексы (индексы, в которых значение ключа состоит из нескольких столбцов) тоже могут быть уникальными.

По умолчанию индексы не уникальны. Для создания неуникального кластерного индекса по таблице, содержащей строки с одинаковыми значениями ключа, используйте ключевые слова allow_dup_row или ignore_dup_row. См. раздел “Повторяющиеся строки” на стр. 367.

clustered

Означает, что физический порядок строк на текущем устройстве хранения базы данных совпадает с порядком строк в индексе. Нижний , или **листовой** уровень кластерного индекса содержит реальные страницы данных. Кластерный индекс практически всегда получает дан-

ные быстрее, чем некластерный индекс. Для каждой таблицы можно создать только один кластерный индекс. См. раздел “Создание кластерных индексов” на стр. 365.

Если ключевое слово clustered не указано, индекс создается с использованием ключевого слова nonclustered.

nonclustered
Означает, что физический порядок строк не совпадает с порядком строк в индексе. Уровень листьев в некластерном индексе содержит указатели на строки на страницах данных. Вы можете создать до 249 некластерных индексов по каждой таблице.

имя_индекса
Имя индекса. Имена индексов должны быть уникальными в пределах таблицы, но могут повторяться в базе данных.

имя_таблицы
Имя таблицы, где находятся индексируемые столбцы. Необходимо указывать имя базы данных, если таблица находится в другой базе данных, и имя владельца, если в базе данных существует несколько таблиц с таким именем. По умолчанию параметры *владелец* и *база_данных* равны текущим пользователю и базе данных.

имя_столбца
Имя столбца или таблицы, по которым строится индекс. Составные индексы могут включать до 16 столбцов. Суммарная максимальная длина всех столбцов в составном индексе не может превышать 600 байт. Столбцы составного индекса указываются в круглых скобках (в том порядке, в котором они должны сортироваться) вслед за параметром *table_name*.

asc | desc
При создании индекса определяет порядок следования его значений (по возрастанию или по убыванию) для определенного столбца. По умолчанию используется сортировка по возрастанию.

fillfactor
Коэффициент заполнения. Определяет, на сколько процентов Adaptive Server заполняет каждую страницу при создании нового индекса по существующим данным. Значение параметра fillfactor действует только при создании индекса. По мере обновления данных степень заполнения страниц не сохраняется.

Указанное значение не сохраняется в таблице sysindexes для отображения процедурой sp_helpindex и последующего использования командой reorg. Для сохранения значения fillfactor используется процедура sp_chgattribute.

По умолчанию значение fillfactor равно 0; оно используется, если в оператор create index не включен параметр with fillfactor (если только это значение не было изменено процедурой sp_configure). В качестве параметра fillfactor можно указывать значение в интервале от 1 до 100.

Если параметр fillfactor равен 0, то создается кластерный индекс с целиком заполненными страницами или некластерный индекс с целиком заполненными листовыми страницами. Такое значение оставляет достаточно места в B-дереве кластерного и некластерного индексов. Чаще всего нет необходимости менять параметр fillfactor.

Если параметру fillfactor присвоено значение 100, то каждая страница создаваемых кластерных и некластерных индексов будет заполнена на 100 процентов. Такое значение параметра fillfactor подходит только для таблиц, предназначенных только для чтения (так как в них никогда не будут добавляться данные).

При значениях fillfactor меньше 100 (кроме 0) новые индексы будут создаваться с частично заполненными страницами. Значение fillfactor, равное 10, может быть оправдано при создании индекса для таблицы, которая со временем будет содержать гораздо больше данных, чем в настоящее время. Однако при малых значениях fillfactor для хранения каждого индекса (или индекса и данных) потребуется больше пространства.

Предупреждение. Параметр fillfactor, указанный при создании кластерного индекса, влияет на объем пространства, который будут занимать данные, поскольку при создании кластерного индекса происходит перераспределение данных.

---

max_rows_per_page

Ограничивает количество строк на страницах данных и листовых страницах индекса. Параметры max_rows_per_page и fillfactor – взаимоисключающие. В отличие от fillfactor, значение параметра max_rows_per_page ограничивает количество строк на странице до тех пор, пока оно не будет изменено процедурой sp_chgattribute.

Если значение max_rows_per_page не указано, при создании таблицы будет использоваться значение 0. Для таблиц и кластерных индексов параметр может принимать значение от 0 до 256. Максимальное количество строк на странице некластерного индекса зависит от размера ключа индекса. Adaptive Server возвращает сообщение об ошибке, если указанное значение превышает максимально допустимую величину.

Если значение параметра max_rows_per_page равно 0, все страницы кластерного индекса и листовые страницы некластерного индекса заполняются полностью. При этом в B-дереве кластерного и некластерного индекса остается достаточно места.

Если параметр max_rows_per_page равен 1, то листовые страницы как кластерных, так и некластерных индексов будут содержать по одной строке. Небольшие значения позволяют снизить количество конфликтов блокировок для часто используемых данных. Однако в этом случае страницы создаваемого индекса будут заняты не полностью, в результате чего увеличивается объем занимаемого пространства и вероятность расщепления страниц.

Если включены службы Component Integration Services, параметр max_rows_per_page нельзя использовать для удаленных серверов.

Предупреждение. Значение параметра max_rows_per_page может влиять на размер пространства, которое занимают данные, так как Adaptive Server перераспределяет их при создании кластерного индекса.

---

with reservepagegap = количество_страниц

Указывает отношение количества заполненных страниц к количеству страниц, которые необходимо оставить пустыми во время будущих операций выделения экстентов с использованием их ввода-вывода. Для будущего расширения индекса будет оставлено по одной пустой странице на заданное *количество_страниц*. Параметр может принимать значения от 0 до 255. Значение по умолчанию – 0.

ignore_dup_key

Отменяет попытку ввести в таблицу с уникальным индексом (кластерным или некластерным) значение ключа, которое уже есть в этой таблице. Adaptive Server отменяет попытку выполнения таких команд, как insert или update, и возвращает информационное сообщение. После того как команда была отменена, содержащая ее транзакция продолжается.

Уникальный индекс нельзя построить по столбцу, содержащему повторяющиеся значения или несколько значений NULL, независимо от того, указан ли параметр ignore_dup_key. При попытке сделать это Adaptive Server выведет сообщение об ошибке, содержащее первое повторяющееся значение. Нужно устранить повторяющиеся значения, чтобы можно было построить уникальный индекс по столбцу.

ignore_dup_row
: Разрешает создать неуникальный кластерный индекс по таблице, содержащей повторяющиеся строки. Если указан параметр ignore_dup_row, то повторяющиеся строки будут удалены из таблицы. Кроме того, любая команда insert или update, пытающаяся создать повторяющиеся строки, будет отменена (без отката всей транзакции). Подробности см. в разделе "Повторяющиеся строки" на стр. 367.

allow_dup_row
: Разрешает создавать неуникальный кластерный индекс по таблице, содержащей повторяющиеся строки, и выполнять команды update и insert, создающие повторяющиеся строки. Подробное описание этих параметров см. в разделе "Повторяющиеся строки" на стр. 367.

sorted_data
: Ускоряет создание кластерных индексов или уникальных некластерных индексов, когда данные в таблице уже отсортированы (например, когда для копирования отсортированных данных в пустую таблицу использовалась утилита bcp). Подробности см. в разделе "Использование параметра sorted_data для ускорения сортировки" на стр. 368.

with statistics using количество_шагов values
: Определяет количество шагов гистограммы, используемой для оптимизации запросов. Если эта инструкция не указана, произойдет следующее:

  - Если гистограмма для первого столбца индекса отсутствует, значение по умолчанию – 20.
  - Если гистограмма для первого столбца индекса уже существует, используется текущее количество шагов.

  Если в качестве *количества_шагов* указано 0, индекс создается заново, но его статистика не переписывается в системных таблицах.

on имя_сегмента
: Создает индекс в указанном сегменте. Чтобы можно было указывать инструкцию on имя_сегмента в команде create index, соответствующее устройство устройство должно быть инициализировано командой disk init, а этот сегмент добавлен в базу данных процедурой sp_addsegment. Список имен сегментов базы данных можно узнать у системного администратора или с помощью процедуры sp_helpsegment.

with consumers

Определяет количество процессов-потребителей, которые должны выполнять сортировку для создания индекса. Реальное количество процессов-потребителей, используемых для сортировки индекса, может быть меньше указанного значения, если во время ее выполнения в Adaptive Server не хватает рабочих процессов.

Примеры

Пример 1. Создание индекса au_id_ind по столбцу au_id таблицы authors:

```
create index au_id_ind on authors (au_id)
```

Пример 2. Создание уникального кластерного индекса au_id_ind по столбцу au_id в таблице authors:

```
create unique clustered index au_id_ind
on authors(au_id)
```

Пример 3. Создание индекса ind1 по столбцам au_id и title_id в таблице titleauthor:

```
create index ind1 on titleauthor (au_id, title_id)
```

Пример 4. Создание некластерного индекса zip_ind по столбцу zip в таблице authors. Каждая страница индекса заполняется на одну четверть, а сортировка ограничивается четырьмя процессами-потребителями:

```
create nonclustered index zip_ind
on authors(postalcode)
with fillfactor = 25, consumers = 4
```

Пример 5. Создание индекса, упорядоченного по возрастанию значений столбца pub_id и убыванию значений столбца pubdate:

```
create index pub_dates_ix
on titles (pub_id asc, pubdate desc)
```

Пример 6. Создание индекса по столбцу title_id. Используются 50 ячеек гистограммы для статистики оптимизатора. На каждые 40 страниц индекса оставляется одна пустая страница:

```
create index title_id_ix
on titles (title_id)
with reservepagegap = 40,
statistics using 50 values
```

Использование

- Периодически выполняйте команду update statistics при добавлении в таблицу данных, изменяющих распределение ключей в индексе. Оптимизатор запросов использует информацию, созданную командой update statistics, для выбора лучшего плана выполнения запроса к таблице.

- Если таблица уже содержит данные, когда вы создаете некластерный индекс, Adaptive Server выполняет команду update statistics для нового индекса. Если таблица уже содержит данные, когда вы создаете кластерный индекс, Adaptive Server выполняет команду update statistics для всех индексов таблицы.
- Индексы следует построить по всем столбцам, которые регулярно используются в соединениях.
- Когда службы Component Integration Services включены, команда create index перестраивается и передается непосредственно на сервер Adaptive Server, связанный с таблицей.

Ограничения

- Нельзя создать индекс по столбцам типов bit, text и image.
- Чтобы создать в таблице столбцы, по которым нельзя построить индекс, нужно указать длину столбца, превышающую максимальную длину строки индекса:

Размер логической страницы	Ограничение на размер индексируемого столбца
2 КБ	600
4 КБ	1250
8 КБ	2600
16 КБ	5300

- По каждой таблице можно построить не более 249 некластерных индексов.
- По одной таблице можно построить не более одного кластерного индекса.
- Ключ индекса может состоять не более чем из 31 столбца (в предыдущих версиях это верхнее ограничение было равно 16). Максимальный размер не должен превышать ограничений, указанных в представленной выше таблице.
- По временной таблице также может быть создан индекс. Он будет удален, когда таблица перестанет существовать.
- Вы можете создать индекс по таблице из другой базы данных, если вы – владелец этой таблицы.
- Нельзя создать индекс по представлению.

- Команда create index работает медленнее, если в это же время выполняется команда dump database.
- Вы можете создать кластерный индекс по секционированной таблице или разбить таблицу на секции с помощью кластерного индекса, если выполняются сразу все следующие условия:
  - включен параметр базы данных select into/bulkcopy/pllsort;
  - Adaptive Server поддерживает параллельную обработку;
  - количество доступных рабочих процессов на единицу больше, чем количество секций.

  Дополнительную информацию о кластерных индексах по секционированным таблицам см. в главе 24, "Параллельная сортировка" книги *Руководство по настройке производительности*.
- По таблице с блокировкой только данных с кластерным индексом можно создать до 249 индексов. По любой такой таблице можно построить один кластерный индекс и 248 некластерных.

Эффективное создание индексов

- Индексы ускоряют поиск данных, но могут замедлить их обновление. Для улучшения производительности следует создавать таблицу на одном сегменте, а некластерные индексы – на другом, когда эти сегменты находятся на разных физических устройствах.
- Adaptive Server может создавать индексы параллельно, если таблица секционирована и сервер поддерживает параллелизм. Кроме того, он может использовать буферы сортировки для сокращения количества операций ввода-вывода, выполняемых во время сортировки. Дополнительную информацию см. в главе, 24 "Параллельная сортировка" книги *Руководство по настройке производительности*.
- Некластерные индексы автоматически перестраиваются при создании кластерного индекса, поэтому их следует строить только после создания кластерного индекса.
- При использовании параллельной сортировки для таблиц с блокировкой только данных количество рабочих процессов должно равняться или превышать количество секций даже для пустых таблиц. Параметр базы данных select into/bulkcopy/pllsort должен быть включен.

Создание кластерных индексов

- Таблица "следует" за кластерным индексом. Когда вы создаете таблицу, используйте команду create clustered index с параметром on имя_сегмента для создания кластерного индекса. При этом таблица перемещается в тот сегмент, где создается индекс.

Если при создании таблицы вы указываете сегмент, а при создании кластерного индекса не указываете, Adaptive Server перемещает таблицу на сегмент по умолчанию, когда он создает там кластерный индекс.

Так как текстовые данные и изображения хранятся на отдельной цепочке страниц, при создании кластерного индекса с указанием параметра on имя_сегмента перемещения столбцов этих типов не происходит.

- Чтобы создать кластерный индекс Adaptive Server дублирует существующие данные. По завершении создания индекса исходные данные удаляются. Перед созданием кластерного индекса убедитесь (с помощью процедуры sp_spaceused), что свободное пространство в базе данных составляет минимум 120% от размера таблицы.
- Кластерный индекс часто создается по первичному ключу таблицы (столбец или столбцы, уникальным образом идентифицирующие строку таблицы). Первичный ключ можно записать в базу данных (для использования в клиентских программах и процедуре sp_depends) процедурой sp_primarykey.
- Чтобы разрешить повторяющиеся значения в кластерном индексе, укажите ключевое слово allow_dup_row.

Определение возрастающего или убывающего порядка сортировки в индексах

- Ключевые слова asc и desc после имен столбцов индекса позволяют указать порядок сортировки для ключа индекса. Если в инструкции order by запроса столбцы указаны в том же порядке, в котором они были указаны при создании какого-либо существующего индекса, то при обработке этого запроса можно будет обойтись без сортировки. Дополнительную информацию см. в главе 8, “Индексирование для повышения производительности”, книги *Руководство по настройке производительности*.

Требования к пространству для индексов

- Пространство под таблицы и индексы выделяется с приращением, равным одному экстенту, или восьми страницам. Каждый раз, когда экстент заполняется, выделяется следующий экстент. Размер пространства, выделенного индексу и используемого им, можно узнать с помощью процедуры sp_spaceused.
- Иногда использование параметра sorted_data позволяет не выполнять копирование строк данных (см. таблицу 7-8 на стр. 369). В этом случае требуется только пространство для структуры индекса. Размер требуемого пространства обычно равен 20% от размера таблицы, хотя и зависит от размера ключа.

Повторяющиеся строки

- Параметры ignore_dup_row и allow_dup_row не имеют значения при создании неуникального некластерного индекса. Adaptive Server присваивает каждой строке в некластерном индексе уникальный внутренний идентификационный номер, поэтому повторяющиеся строки не являются проблемой даже при наличии одинаковых значений данных.
- Параметры ignore_dup_row и allow_dup_row – взаимоисключающие.
- Неуникальный кластерный индекс допускает одинаковые ключи, но не допускает одинаковых строк, если не указан параметр allow_dup_row.
- Параметр allow_dup_row позволяет создать неуникальный кластерный индекс по таблице, содержащей одинаковые строки. Если у таблицы есть неуникальный кластерный индекс, созданный без использования параметра allow_dup_row, нельзя создать новые одинаковых строки командами insert или update.

  Если какой-либо индекс в таблице уникален, требование уникальности имеет приоритет перед параметром allow_dup_row. Нельзя создать индекс с параметром allow_dup_row, если по какому-либо столбцу таблицы уже построен уникальный индекс.
- Параметр ignore_dup_row также используется при создании неуникального кластерного индекса. Параметр ignore_dup_row удаляет повторения из пакета данных. Этот параметр отменяет команды insert и update, пытающиеся создать повторяющиеся строки, но не вызывает отката всей транзакции.
- Таблица 7-6 иллюстрирует влияние параметров allow_dup_row и ignore_dup_row на попытки создания неуникального кластерного индекса по таблице, содержащей повторяющиеся строки, и последующего создания повторяющихся строк в таблице.

Таблица 7-6. Параметры ignore_dup_row и allow_dup_row для неуникальных кластерных индексов

Значение параметра	Создание индекса по таблице, содержащей повторяющиеся строки	Вставка повторяющихся строк в таблицу с индексом
Ни один параметр не указан	Команда create index не будет выполнена.	Команда insert не будет выполнена.
Указан параметр allow_dup_row	Команда create index успешно выполняется.	Команда insert успешно выполняется.
Указан параметр ignore_dup_row	Индекс создается, но повторяющиеся строки удаляются и выводится сообщение об ошибке.	Вставляются все неповторяющиеся строки и выводится сообщение об ошибке.

Таблица 7-7 иллюстрирует, какие параметры можно использовать с различными типами индексов:

Таблица 7-7. Параметры индекса

Тип индекса	Параметры
Кластерный	ignore_dup_row | allow_dup_row
Уникальный кластерный	ignore_dup_key
Некластерный	Нет
Уникальный некластерный	ignore_dup_key, ignore_dup_row

Использование уникальных ограничений вместо индексов

- Уникальные индексы можно создать не только с помощью команды create index, но и неявно, указав уникальные ограничения в командах create table или alter table. Уникальное ограничение создает кластерный или некластерный индекс по столбцам таблицы. Эти *неявные* индексы называются по имени соответствующего ограничения и следуют тем же правилам, что и индексы, созданные командой create index.

- Индексы, образованные уникальными ограничениями, нельзя удалить оператором drop index. Они удаляются только вместе с таблицей или ограничением, которое можно удалить оператором alter table. Более подробная информация об уникальных ограничениях содержится в описании оператора create table.

Использование параметра *sorted_data* для ускорения сортировки

- Параметр sorted_data позволяет ускорить создание индекса, пропустив этап сортировки и в некоторых случаях избежав необходимости копирования строк данных на новые страницы. Увеличение скорости становится значительным при работе с большими таблицами, а для таблиц размером больше 1 ГБ скорость может увеличиваться в несколько раз.

  Если параметр sorted_data указан, но данные не отсортированы, Adaptive Server отображает сообщение об ошибке и команда не выполняется.

  Создание неуникального некластерного индекса происходит успешно, если не существует строк с повторяющимися ключами. Если строки с повторяющимися ключами существуют, Adaptive Server отображает сообщение об ошибке и команда не выполняется.

- Последствия указания параметра sorted_data при создании кластерного индекса зависят от того, какие другие параметры были указаны в команде create index и от того, секционирована ли таблица. Некоторые параметры требуют копирования данных (если вообще ис-

пользуются) для несекционированных таблиц и сортировки вместе с копированием данных для секционированных таблиц, другие параметры требуют копирования данных, только если:

- указан параметр ignore_dup_row;
- указан параметр fillfactor;
- указана инструкция on имя_сегмента с сегментом, отличным от того, где расположены данные таблицы;
- в инструкции max_rows_per_page указано значение, отличное от связанного с таблицей.

- В таблице 7-8 перечислены ситуации, в которых требуется сортировка и копирование секционированной и несекционированной таблицы.

Таблица 7-8. Использование параметра sorted_data при создании кластерного индекса

Параметры	Секционированная таблица	Несекционированная таблица
Параметры не заданы	Параллельная сортировка, копирование данных с равномерным распределением по секциям, создание дерева индекса.	Параллельная или непараллельная сортировка, копирование данных, создание дерева индекса.
Только with sorted_data или with sorted_data on тот_же_сегмент	Создается только дерево индекса. Сортировка и копирование данных не выполняются. Параллелизм не поддерживается.	Создается только дерево индекса. Сортировка и копирование данных не выполняются. Параллелизм не поддерживается.
with sorted_data и ignore_dup_row или fillfactor или on другой_сегмент или max_rows_per_page	Параллельная сортировка, копирование данных с равномерным распределением по секциям, создание дерева индекса.	Копирование данных и создание дерева индекса. Сортировка не выполняется. Параллелизм не поддерживается.

Определение количества шагов гистограммы

- Инструкция with statistics позволяет задать количество шагов гистограммы для первого столбца индекса. Гистограммы используются при оптимизации запросов для определения количества строк, соответствующих аргументам поиска для определенного столбца.
- Чтобы повторно создать индекс без обновления значений в таблице sysstatistics для столбца, укажите количество шагов, равное 0. Это позволяет избежать перезаписи статистики, которая была изменена утилитой optdiag.

Свойства управления пространством

- Параметры fillfactor, max_rows_per_page и reservepagegap помогают управлять пространством на страницах индекса разными способами.
  - Параметр fillfactor применяется к индексам для любых схем блокировки. В кластерных индексах по таблицам с блокировкой всех страниц он влияет на страницы данных в таблице. В остальных индексах он влияет на листовой уровень индекса.
  - Параметр max_rows_per_page применяется только к страницам индексов для таблиц с блокировкой всех страниц.
  - Параметр reservepagegap применяется к таблицам и индексам для любых схем блокировки.
- Параметр reservepagegap влияет на использование пространства в индексе в следующих случаях:
  - при создании индекса;
  - при выполнении над индексом команд reorg;
  - при перестройке некластерных индексов после создания кластерного индекса.
- Если значение параметра reservepagegap указано в команде create clustered index, оно применяется:
  - к данным и страницам индекса таблицы с блокировкой всех страниц;
  - только к страницам индекса таблиц с блокировкой только данных.
- Параметр *количество_страниц* определяет отношение заполненных страниц к пустым страницам на уровне листьев индекса. Поэтому индекс может выделять пространство рядом с существующими страницами по мере необходимости. Например, если параметр reservepagegap равен 10, пустая страница оставляется для каждых девяти заполненных страниц.
- Параметр reservepagegap, указанный в команде create clustered index, создающей индекс по таблице с блокировкой всех страниц, переписывает любое значение, заданное ранее операторами create table или alter table.
- Свойства управления пространством индекса можно изменить процедурой sp_chgattribute. Изменение свойств процедурой sp_chgattribute не оказывает немедленного влияния на простран-

ство индекса по таблице. Для выделения большого пространства в будущем, например, командой reorg rebuild, используйте процедуру sp_chgattribute.

- Значение параметра fillfactor, заданное процедурой sp_chgattribute, хранится в столбце fill_factor таблицы sysindexes. Параметр fillfactor применяется при повторном создании индекса, вызванном выполнением команды alter table...lock или reorg rebuild.

Параметры индекса и режимы блокировки

- В таблице 7-9 перечислены все параметры, которые можно использовать при создании индексов по таблицам с блокировкой только данных и с блокировкой всех страниц. Если таблица использует блокировку только данных, то параметры ignore_dup_row и allow_dup_row действуют при создании индекса по этой таблице, но не действуют при операциях insert и update над этой таблицей. Таблицы с блокировкой только данных всегда позволяют вставлять повторяющиеся строки.

Таблица 7-9. Параметры оператора create index, поддерживаемые схемами блокировки

Тип индекса	Таблица с блокировкой всех страниц	Таблица с блокировкой только данных	
		Во время создания индекса	Во время вставки
Кластерный	allow_dup_row, ignore_dup_row	allow_dup_row, ignore_dup_row	allow_dup_row
Уникальный кластерный	ignore_dup_key	ignore_dup_key	ignore_dup_key
Некластерный	Нет	Нет	Нет
Уникальный некластерный	ignore_dup_key	ignore_dup_key	ignore_dup_key

В таблице 7-10 описаны результаты выполнения команд, пытающихся вставить повторяющиеся строки в таблицы с кластерными индексами, а также удалить или повторно создать кластерные индексы.

Таблица 7-10. Результаты применения параметров allow_dup_row и ignore_dup_row

Параметры	Таблица с блокировкой всех страниц	Таблица с блокировкой только данных
Параметры не заданы	При попытке вставки генерируется сообщение об ошибке 2615. Повторное создание индекса выполняется успешно.	Вставка выполняется успешно. При попытке повторного создания индекса генерируется сообщение об ошибке 1508.

Параметры	Таблица с блокировкой всех страниц	Таблица с блокировкой только данных
allow_dup_row	Вставка и повторное создание индекса выполняются успешно.	Вставка и повторное создание индекса выполняются успешно.
ignore_dup_row	При попытке вставки генерируется сообщение об ошибке “Duplicate row was ignored”. Повторное создание индекса выполняется успешно.	Вставка выполняется успешно. При повторном создании индекса повторяющиеся строки удаляются.

Использование параметра *sorted_data* при создании индексов по таблицам с блокировкой только данных

- Параметр sorted_data в операторе create index можно использовать только сразу после операции массового копирования в пустую таблицу. Если изменение данных в таблице вызвало выделение новых страниц, параметр sorted_data использовать нельзя.
- Другие значения свойств управления пространством могут отменить действие параметра sorted_data, дающего возможность избежать сортировки.

Получение информации о таблицах и индексах

- Каждому индексу, в том числе и составному, соответствует одна строка в таблице sysindexes.
- Информацию о порядке данных, извлекаемых через индексы, и о влиянии порядка сортировки, установленного в Adaptive Server, см. в разделе order by.
- Для просмотра информации об индексах, построенных по таблице, используется процедура sp_helpindex.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на выполнение команды create index принадлежат владельцу таблицы и не могут быть переданы другим пользователям.

См. также

Команды alter table, create table, drop index, insert, order by, set, update

Системные процедуры sp_addsegment, sp_chgattribute, sp_helpindex, sp_helpsegment, sp_spaceused

Утилиты optdiag

create plan

Описание

Создает абстрактный план.

Синтаксис

create plan *запрос план*
[into *имя_группы*]
[and set @*новый_идентификатор*]

Параметры

запрос
Строковый литерал, параметр или локальная переменная с текстом запроса на языке SQL.

план
Строковый литерал, параметр или локальная переменная с выражением абстрактного плана.

into имя_группы
Определяет имя группы абстрактных планов.

and set @новый_идентификатор
Возвращает идентификационный номер абстрактного плана в переменную.

Примеры

Пример 1. Создание абстрактного плана для указанного запроса:

```
create plan "select * from titles where price > $20"
"(t_scan titles)"
```

Пример 2. Создание абстрактного плана для запроса в группе dev_plans . Идентификатор плана возвращается в переменную *@id*:

```
declare @id int
create plan "select au_fname, au_lname from authors
where au_id = '724-08-9931' "
"(i_scan au_id_ix authors)"
into dev_plans
and set @id
select @id
```

Использование

- Команда create plan сохраняет абстрактный план в группу, заданную ключевым словом into. Если группа не указана, план сохраняется в активную на текущий момент группу.
- Запросы и абстрактные планы, указанные в команде create plan, не проверяются на соотвествие синтаксису SQL и синтаксису абстрактных планов. Совместимость плана с текстом SQL также не проверяется. Все планы, созданные командой create plan, следует немедленно проверить на правильность, выполнив запрос, указанный в этой команде.

- Если в группе уже есть план запроса с тем же текстом SQL, необходимо включить режим replace командой set plan replace on. Иначе команда create plan не будет выполнена.
- Переменную *@new_id* необходимо объявить перед использованием в инструкции and set.
- Группа абстрактных планов, указанная после ключевого слова into, должна существовать.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду create plan может выполнить любой пользователь. На выполнение этой команды полномочий не требуется.

См. также

Команды set plan

Системные процедуры sp_add_qpgroup, sp_find_qplan, sp_help_qpgroup, sp_set_qplan

create procedure

Описание

Создает хранимую процедуру или расширенную хранимую процедуру (extended stored procedure, ESP) с одним или несколькими параметрами, определяемыми пользователями.

Примечание. Синтаксис и способы использования команды SQLJ для создания процедур содержатся в разделе create function (SQLJ) на стр. 355.

Синтаксис

```
create procedure [владелец.]имя_процедуры[;количество]
        [[(]@имя_параметра
            тип_данных [(длина ) | (точность  [, масштаб ])]
            [= значение_по_умолчанию][output]
        [, @имя_параметра
            тип_данных [(длина ) | (точность  [, масштаб ])]
            [=значение_по_умолчанию][output]]...[)]]
        [with recompile]
        as {SQL_операторы | external name имя_dll}
```

Параметры

имя_процедуры

Имя процедуры. Должно быть допустимым идентификатором, не может быть переменной. Чтобы создать процедуру с тем же именем, что и процедура, принадлежащая другому пользователю в текущей базе данных, перед именем процедуры нужно указать имя владельца. По умолчанию *владелец* – текущий пользователь.

;количество

Необязательный целочисленный параметр, позволяющий объединить процедуры с одинаковым именем в группу, чтобы их можно было удалить все вместе одним оператором drop procedure. Таким способом часто группируются процедуры, используемые в одном приложении. Например, если процедуры, используемые приложением orders, имеют имена orderproc;1, orderproc;2 и т.д., следующий оператор удаляет всю группу:

```
drop proc orderproc
```

После того как процедуры были сгруппированы, отдельную процедуру, принадлежащую группе процедур, нельзя удалить. Например, следующий оператор недопустим:

```
drop procedure orderproc;2
```

Процедуры нельзя группировать, если Adaptive Server работает в **опробованной конфигурации**. Опробованная конфигурация запрещает группировку процедур и требует, чтобы каждая хранимая процедура имела уникальный идентификатор объекта и могла быть удалена от-

дельно. Администратор безопасности системы может запретить группировку процедур, изменив значение параметра allow procedure grouping с помощью системной процедуры sp_configure. Информацию по опробованной конфигурации см. в книге *Руководство по системному администрированию*.

имя_параметра

Имя аргумента процедуры. Значения параметров указываются при выполнении процедуры. Имена параметров необязательно указывать в команде create procedure: процедура может не иметь аргументов.

Имена параметров должны быть допустимыми идентификаторами и начинаться с символа @. Максимальная длина имени параметра, включая знак @, составляет 30 символов. Параметры являются локальными по отношению к процедуре: другие процедуры могут использовать параметры с такими же именами.

Если значение параметра содержит не буквенно-цифровые символы, оно должно быть заключено в кавычки. Это правило распространяется на имена объектов, уточненные именем базы данных или владельца (так как они содержат точку). Значение параметра, начинающееся с цифры, также нужно заключить в кавычки.

тип_данных[(*длина*) | (*точность* [, *масштаб*])]

Тип данных параметра. Подробности о типах данных см. в главе 1, "Системные и пользовательские типы данных". Параметры хранимой процедуры не могут иметь тип text, image и пользовательский тип, базовым типом которого является text или image.

Вместе с типами данных char, varchar, unichar, univarchar, nchar, nvarchar, binary и varbinary указывается *длина* (в круглых скобках). Если длина не указана, значение параметра усекается до одного символа.

Вместе с типом данных float в круглых скобках можно указать двоичную *точность*. Если точность не указана, Adaptive Server использует точность, заданную по умолчанию для данной платформы.

Вместе с типами данных numeric и decimal в круглых скобках через запятую можно указать *точность* и *масштаб*. Если точность и масштаб не указаны, Adaptive Server использует значения по умолчанию – 18 для точности и 0 для масштаба.

значение_по_умолчанию

Значение по умолчанию для параметра процедуры. Если значение по умолчанию задано, пользователь может не указывать значение параметра при вызове процедуры. Значение по умолчанию должно

быть константой. Оно может содержать метасимволы (%, _, [ ] и [^]), если параметр используется в процедуре вместе с ключевым словом like (см. пример 2).

Значение по умолчанию может равняться NULL. В определении процедуры можно указать, что некоторое действие должно быть выполнено, если значение параметра равно NULL (см. пример 3).

output
Указывает, что параметр является выходным. Значение этого параметра может быть возвращено команде execute, вызывающей процедуру. Выходные параметры используются для возврата информации в вызывающую процедуру (см. пример 5).

Чтобы значение параметра можно было передать вверх на несколько уровней вложенности, в каждой процедуре должен быть определен соответствующий выходной параметр (в том числе и в команде execute, вызвающей процедуру высшего уровня).

Можно использовать сокращенную форму ключевого слова output – out.

with recompile
Означает, что план для процедуры никогда не будет сохраняться, а при каждом ее выполнении будет создаваться новый план. Эту необязательную инструкцию следует указать, если вы считаете, что выполнение процедуры будет нетипичным, то есть когда необходимо создать новый план. Инструкция with recompile не влияет на выполнение расширенной хранимой процедуры.

операторы_SQL
Действия, выполняемые процедурой. Этот параметр может содержать неограниченное количество любых операторов SQL, кроме create view, create default, create rule, create procedure, create trigger и use.

SQL-операторы команды create procedure часто содержат управляющие конструкции, в том числе declare, if...else; while, break, continue, begin...end, goto label, return, waitfor и /* комментарии */. В них также могут быть указаны параметры, объявленные в процедуре.

SQL-операторы могут ссылаться на объекты в другой базе данных (при этом имена этих объектов должны быть соответсвующим образом уточнены именем базы данных).

external name
Создает расширенную хранимую процедуру. Если используется синтаксис as external name, то процедуру нельзя группировать (указывать параметр *номер*).

имя_dll

Определяет имя динамической библиотеки (dynamic link library, DLL) или разделяемой библиотеки, содержащей функции, реализующие расширенную хранимую процедуру. *Имя_dll* может быть указано без расширения или с расширением, характерным для данной платформы, например *.dll* в Windows NT или *.so* в Sun Solaris. Если расширение указано, *имя_dll* должно быть заключено в кавычки.

Примеры

Пример 1. Процедура showind отображает имя таблицы, переданное ей в качестве параметра, и имена и идентификаторы всех индексов, построенных по ее столбцам:

```
create procedure showind @tabname varchar(30)
as
  select sysobjects.name, sysindexes.name, indid
  from sysindexes, sysobjects
  where sysobjects.name = @tabname
  and sysobjects.id = sysindexes.id
```

Ниже перечислены допустимые синтаксисы для выполнения процедуры showind:

```
execute showind titles
execute showind @tabname = "titles"
```

или, если это первый оператор в файле или пакете:

```
showind titles
```

Пример 2. Следующая процедура отображает информацию о системных таблицах, если параметр не указан:

```
create procedure
showsysind @table varchar(30) = "sys%"
as
  select sysobjects.name, sysindexes.name, indid
  from sysindexes, sysobjects
  where sysobjects.name like @table
  and sysobjects.id = sysindexes.id
```

Пример 3. В следующей процедуре указаны действия, которые необходимо выполнить, если параметр равен NULL (то есть параметр не указан пользователем):

```
create procedure
showindnew @table varchar(30) = null
as
  if @table is null
   print "Please give a table name"
```

```
else
 select sysobjects.name, sysindexes.name, indid
 from sysindexes, sysobjects
 where sysobjects.name = @table
 and sysobjects.id = sysindexes.id
```

Пример 4. Следующая процедура вычисляет произведение двух целочисленных параметров и возвращает результат в выходной параметр *@result*:

```
create procedure mathtutor @mult1 int, @mult2 int,
  @result int output
as
select @result = @mult1 * @mult2
```

Если в процедуру передаются три числа, оператор select выполняет умножение и присваивает значение, но не отображает выходной параметр:

```
mathtutor 5, 6, 32
(return status 0)
```

Пример 5. В этом примере и процедура, и оператор execute сопровождаются ключевым словом `output` с именем параметра, возвращающим значение вызывающей процедуре:

```
declare @guess int
select @guess = 32
exec mathtutor 5, 6, @result = @guess output

(1 row affected)
(return status = 0)

Return parameters:

@result
-----------
         30
```

Выходной параметр *@result* и все последующие параметры оператора execute *должны* передаваться в форме:

```
@параметр = значение
```

- Значение возвращаемого параметра отображается всегда, независимо от того, изменилось оно или нет.
- *@result* необязательно объявлять в вызывающем пакете, поскольку это имя параметра, передаваемого в `mathtutor`.

- Хотя значение параметра *@result* возвращается в переменную, указанную в операторе execute (в нашем случае – *@guess*), он отображается под собственным заголовком (*@result*).

Пример 6. Возвращаемые параметры можно использовать в других SQL-операторах пакета или вызывающей процедуры. Следующий пример демонстрирует, как можно использовать значение переменной *@guess* в условных инструкциях после оператора execute, сохраняя его в другой переменной *@store* во время вызова процедуры. Если возвращаемые параметры используются в операторе execute, который является частью SQL-пакета, возвращаемые значения печатаются под собственным заголовком перед последующими операторами пакета.

```
declare @guess int
declare @store int
select @guess = 32
select @store = @guess
execute mathtutor 5, 6, @result = @guess output
select Your_answer = @store, Right_answer = @guess
if @guess = @store
    print "Right-o"
else
    print "Wrong, wrong, wrong!"

(1 row affected)
(1 row affected)
(return status = 0)

Return parameters:

@result
-----------
         30
Your_answer Right_answer
----------- ------------
         32           30

(1 row affected)
Wrong, wrong, wrong!
```

Пример 7. Создание расширенной хранимой процедуры xp_echo, которая присваивает значение входного параметра *@in* выходному параметру *@out*. Код этой процедуры содержится в функции xp_echo, которая находится в динамической библиотеке *sqlsrvdll.dll*:

```
create procedure xp_echo @in varchar(255),
        @out varchar(255) output
as external name "sqlsrvdll.dll"
```

Использование

- Созданную процедуру можно выполнить оператором execute, указав ее имя и параметры. Если процедура является первым оператором в пакете, ее имя можно задать без указания ключевого слова execute.
- Системная процедура sp_hidetext позволяет скрыть исходный текст процедуры, хранящийся в таблице syscomments.
- Если пакет хранимой процедуры выполняется успешно, Adaptive Server присваивает глобальной переменной *@@error* значение 0.

Ограничения

- Максимальное количество параметров хранимой процедуры равно 255.
- Максимальное количество локальных и глобальных переменных в процедуре ограничено только объемом доступной памяти.
- Максимальный объем текста хранимой процедуры составляет 16 МБ.
- Оператор create procedure нельзя объединить с другими операторами в один пакет.
- Хранимую процедуру можно создать только в текущей базе данных, хотя она может ссылаться на объекты в других базах данных. Все объекты, на которые ссылается процедура, должны существовать в момент ее создания. На объект, созданный внутри процедуры, можно сослаться при условии, что данный объект был создан до ссылки на него.

  Команду alter table нельзя использовать в процедуре для добавления столбца и последующих ссылок на него из этой процедуры.
- Процедура, созданная командой create procedure и использующая оператор select *, не будет выбирать новые добавленные в таблицу столбцы (даже при использовании параметра with recompile команды execute). Необходимо удалить процедуру командой drop и создать ее повторно.

- В хранимой процедуре нельзя создать объект (включая временную таблицу), удалить его, а затем создать новый объект с тем же именем. Adaptive Server создает объект, определенный в хранимой процедуре, во время ее выполнения, а не во время компиляции.

Предупреждение. Некоторые изменения в базе данных, например удаление и повторное создание индексов, могут привести к изменению идентификатора объекта. Когда идентификаторы объектов изменяются, хранимые процедуры автоматически перекомпилируются и могут слегка увеличиться в размере. Оставляйте немного места для этого увеличения.

Расширенные хранимые процедуры

- Если используется синтаксис as external name, команда create procedure регистрирует расширенную хранимую процедуру (ESP). Расширенные хранимые процедуры выполняют функции, написанные на процедурном языке, а не команды Transact-SQL.
- *В Windows NT* ESP-функция не должна вызывать сигнальную процедуру этапа выполнения на языке Си. Это приведет к сбою XP Server, так как Open Server™ не поддерживает обработку сигналов в Windows NT.
- Для поддержания многопоточности ESP-функции должны использовать функцию Open Server srv_yield, которая приостанавливает и повторно планирует работу потока XP Server, позволяя выполниться другому потоку с равным или более высоким приоритетом.
- Механизм поиска файла DLL зависит от платформы. В Windows NT поиск имени файла DLL выполняется в следующей последовательности:

  a каталог, из которого загружается приложение;

  b текущий каталог;

  c системный каталог (SYSTEM32);

  d каталоги, перечисленные в переменной окружения PATH.

  Если DLL не находится в первых трех каталогах, включите путь к нужному каталогу в переменную PATH.

  В UNIX метод поиска зависит от платформы. Если DLL или общую библиотеку найти не удается, проверяется каталог *$SYBASE/lib*.

  Абсолютные имена путей не поддерживаются.

Системные процедуры

- Системные администраторы могут создавать новые системные процедуры в базе данных sybsystemprocs. Имена системных процедур должны начинаться с символов “sp_”. Эти процедуры можно выполнить из любой базы данных, указав только имя; необязательно уточнять имена процедур с помощью имени базы данных sybsystemprocs. Более подробную информацию о создании системных процедур см. в книге *Руководство по системному администрированию*.
- Результаты системных процедур различаются в зависимости от контекста, в котором они выполняются. Например, процедура sp_foo, вызывающая системную функцию db_name(), возвращает имя базы данных, из которой она выполняется. Если ее выполнить из базы данных pubs2, она возвратит значение “pubs2”:

  ```
  use pubs2
  sp_foo
  ------------------------------
  pubs2
  ```

  Если ее выполнить из базы данных sybsystemprocs, она возвратит значение “sybsystemprocs”:

  ```
  use sybsystemprocs
  sp_foo
  ------------------------------
  sybsystemprocs
  ```

Вложенные процедуры

- Вложение процедур происходит, когда одна хранимая процедура вызывает другую.
- Вложенная процедура может обращаться к объектам, созданным вызывающей процедурой.
- Уровень вложенности увеличивается при выполнении вызванной процедуры и уменьшается при ее завершении. Если уровень вложенности превышает 16, транзакция завершается неудачно.
- Процедуру можно вызвать по имени или по имени переменной вместо фактического имени процедуры.
- Текущий уровень вложенности хранится в глобальной переменной *nestlevel*.

Возвращаемое состояние процедуры

- Хранимые процедуры могут возвращать целочисленное значение, называемое *возвращаемым состоянием*. Возвращаемое состояние указывает, что процедура выполнена успешно, или сообщает тип ошибки.
- Когда вы выполняете хранимую процедуру, она автоматически возвращает соответствующий код состояния. В настоящий момент Adaptive Server возвращает следующие коды состояния.

Код	Значение
0	Процедура выполнена без ошибок
-1	Объект отсутствует
-2	Ошибка в типе данных
-3	Процесс подвергся взаимоблокировке
-4	Ошибка, связанная с отсутствием полномочий
-5	Синтаксическая ошибка
-6	Смешанная пользовательская ошибка
-7	Ошибка ресурсов, например нехватка пространства
-8	Нефатальная внутренняя проблема
-9	Достигнут системный предел
-10	Неисправимая внутренняя несогласованность
-11	Неисправимая внутренняя несогласованность
-12	Таблица или индекс повреждены
-13	База данных повреждена
-14	Аппаратная ошибка

  Коды от -15 до -99 зарезервированы для будущего использования.

- Пользователи могут формировать собственные возвращаемые состояния оператором return. Состоянием может быть любое целое число, кроме чисел от 0 до -99. Следующий пример возвращает “1”, если для книги существует законный контракт, и “2” во всех других случаях:

```
create proc checkcontract @titleid tid
as
if (select contract from titles where
        title_id = @titleid) = 1
   return 1
else
   return 2
checkcontract @titleid = "BU1111"
(return status = 1)
checkcontract @titleid = "MC3026"
(return status = 2)
```

- Если во время выполнения возникает сразу несколько ошибок возвращается код с наибольшим абсолютным значением. Пользовательские значения имеют приоритет над системными.

Идентификаторы объектов

- Для переименования хранимой процедуры используется системная процедура sp_rename.
- Чтобы переименовать расширенную хранимую процедуру, нужно удалить ее, переименовать и перекомпилировать базовую функцию и создать процедуру.
- Если процедура ссылается на таблицы, столбцы или представления, имена которых не являются допустимыми идентификаторами, необходимо включить параметр *quoted_identifier* перед выполнением команды *create procedure* и заключить каждое такое имя в двойные кавычки. Параметр *quoted_identifier может* быть выключен при выполнении процедуры.
- Если любой объект, на который ссылается процедура, был переименован, процедуру необходимо удалить и создать заново.
- Имена объектов, указанные в командах create table и dbcc внутри хранимой процедуры, должны содержать имя владельца, если вы хотите, чтобы другие пользователи могли обращаться к данной процедуре. Например, пользователь “mary”, владеющая таблицей marytab, должна полностью указать имя таблицы внутри хранимой процедуры (когда оно используется с этими командами), если она хочет, чтобы другие пользователи могли выполнить хранимую процедуру. Это нужно, потому что имена объектов разрешаются во время выполнения процедуры. Когда другой пользователь попытается выполнить процедуру, Adaptive Server найдет таблицу marytab, владельцем которой является “mary”, а не таблицу marytab, принадлежащую пользователю, выполняющему процедуру.

  Имена объектов, используемых в других операторах (например, select или insert) внутри хранимой процедуры, не нужно указывать полностью, т.к. они разрешаются во время компиляции процедуры.

Временные таблицы и процедуры

- Вы можете создать хранимую процедуру для ссылки на временную таблицу, созданную в текущем сеансе. Временная таблица, созданная внутри хранимой процедуры, удаляется после ее завершения. Подробности см. в книге *Transact-SQL User’s Guide*.
- Такие системные процедуры, как sp_help, работают с временными таблицами, только если они вызваны из базы данных tempdb.

Задание параметров в процедурах

- Команду set можно использовать внутри хранимой процедуры. Большинство параметров команды set остаются действительными во время выполнения процедуры, а затем принимают исходные значения.

  Однако при использовании параметров команды set (таких как identity_insert), которые требуют, чтобы пользователь был владельцем объекта, другие пользователи не смогут выполнить хранимую процедуру.

Получение информации о процедурах

- Для получения отчета об объектах, на которые ссылается процедура, используйте системную процедуру sp_depends.
- Для отображения текста оператора create procedure, который хранится в таблице syscomments, выполните системную процедуру sp_helptext, указав имя нужной процедуры в качестве параметра. При использовании системной процедуры sp_helptext вы должны работать с базой данных, где находится нужная процедура. Для отображения текста системной процедуры выполните процедуру sp_helptext из базы данных sybsystemprocs.
- Для просмотра списка системных расширенных хранимых процедур и динамических библиотек, их поддерживающих, выполните процедуру sp_helpextendedproc из базы данных sybsystemprocs.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на выполнение команды create procedure принадлежат владельцу базы данных, который может передать их другим пользователям.

Полномочия на использование процедуры должны предоставляться явно командой grant и могут отзываться командой revoke.

Полномочия на объекты при создании процедуры. При создании процедуры не происходит проверки полномочий на объекты (например, таблицы и представления), на которые она ссылается. Таким образом, процедура будет успешно создана, даже если у создающего ее пользователя нет доступа к ее объектам. Все полномочия проверяются при выполнении процедуры.

Полномочия на объекты при выполнении процедуры. Когда процедура выполняется, проверка полномочий на доступ к объектам происходит по-разному в зависимости от того, принадлежат ли процедура и все объекты, на которые она ссылается, одному пользователю.

- Если объекты, на которые ссылается процедура, принадлежат разным пользователям, то пользователю, который вызвал процедуру, должен быть предоставлен прямой доступ к этим объектам. Например, если процедура выполняет выборку из таблицы, к которой пользователь, вызывающий процедуру, не имеет доступа, выполнение процедуры завершается ошибкой.
- Если процедура и ее объекты принадлежат одному пользователю, применяются специальные правила. Пользователю, который вызывает процедуру, автоматически назначаются “неявные полномочия” на доступ к объектам процедуры, даже если он не может обращаться к ним напрямую. С помощью хранимых процедур вы можете предоставить ограниченный доступ пользователям, не имеющим прямого доступа к вашим таблицам и представлениям. Таким образом, хранимые процедуры можно использовать в качестве механизма защиты. Например, с помощью хранимой процедуры можно предоставить пользователю доступ только к определенным столбцам и строкам таблицы.

  Подробное описание правил для неявных полномочий см. в книге *Руководство по системному администрированию*.

См. также

Команды begin...end, break, continue, declare, drop procedure, execute, goto label, grant, if...else, return, select, waitfor, while

Системные процедуры sp_addextendedproc, sp_helpextendedproc, sp_helptext, sp_hidetext, sp_rename

create procedure (SQLJ)

Описание

Создает хранимую процедуру SQLJ путем добавления SQL-оболочки к статическому методу Java. Эта процедура может принимать параметры от пользователей и возвращать результирующие наборы и выходные параметры.

Примечание. Синтаксис и способы использования команды Transact-SQL по созданию процедур см. в разделе create procedure на стр. 375.

Синтаксис

```
create procedure [владелец.]имя_процедуры_sql
        ([ [ in | out | inout ] имя_параметра_sql
            тип_данных_sql [( длина) |
            (точность[, масштаб]) ]
        [, [ in | out | inout ] имя_параметра_sql
            тип_данных_sql [( длина) |
            (точность[, масштаб]) ] ]
        ...])
        [modifies sql data ]
        [dynamic result sets целое_число]
        [deterministic | not deterministic]
        language java
        parameter style java
        external name 'имя_метода_java
            [ ( [тип_данных_java[, тип_данных_java
            ...]] ) ]'
```

Параметры

имя_процедуры_sql

Имя процедуры Transact-SQL. Должно быть допустимым идентификатором, не может быть переменной. Чтобы создать другую процедуру с тем же именем, что и существующая процедура, принадлежащая другому пользователю в текущей базе данных, перед именем процедуры нужно указать имя владельца. По умолчанию *владелец* – текущий пользователь.

in | out | inout

Режим указанного параметра: in – входной параметр; out – выходной параметр; inout – параметр, который одновременно является и входным, и выходным. Режим по умолчанию – in.

имя_параметра_sql

Имя аргумента процедуры. Значения входных параметров указываются при вызове процедуры. Параметры указывать не обязательно: хранимая процедура SQLJ может не иметь аргументов.

Имена параметров должны соответствовать правилам для идентификаторов. Если значение параметра содержит не буквенно-цифровые символы, оно должно быть заключено в кавычки. Это правило распространяется на имена объектов, уточняемые именем базы данных или владельца, так как они содержат точку. Значение параметра, начинающееся с цифры, также должно быть заключено в кавычки.

тип_данных_sql [(*длина*) | (*точность* [, *масштаб*])]
Тип данных Transact-SQL для параметра.

тип_данных_sql является сигнатурой процедуры SQL.

modifies sql data
Уазывает, что метод Java вызывает операции SQL, читает и изменяет данные в базе данных. Это единственно возможная настройка, она является настройкой по умолчанию. Она обеспечивает синтаксическую совместимость со стандартом ANSI.

dynamic result sets целое_число
Указывает, что метод Java может возвращать результирующие наборы SQL. *целое_число* определяет максимальное количество результирующих наборов, которые может вернуть метод. Это значение определяется реализацией.

deterministic | not deterministic
Синтаксис поддерживается для совместимости с другими разработчиками программного обеспечения, соответствующего стандарту SQLJ.

language java
Указывает, что внешняя подпрограмма написана на Java. Эта инструкция является обязательной для хранимых процедур SQLJ.

parameter style java
Указывает, что параметры, передаваемые внешней подпрограмме во время выполнения, являются параметрами Java. Эта инструкция является обязательной для хранимых процедур SQLJ.

external
Указывает, что команда create procedure определяет SQL-имя для внешней подпрограммы, написанной на языке, отличном от SQL.

имя
Определяет имя внешней подпрограммы (метода Java). Указанное имя является литералом символьной строки и должно быть заключено в кавычки.

```
'имя_метода_java [тип_данных_java
          [{, тип_данных_java} ...]]'
```

имя_метода_java
Определяет имя внешнего метода Java.

тип_данных_java
Указывает тип данных Java, совместимый в обычном смысле (mappable) или совместимый по набору результатов (result-set mappable). Является сигнатурой метода Java.

Примеры

В этом примере создается SQLJ-процедура java_multiply, которая перемножает два целых числа и возвращает целочисленный результат.

```
create procedure java_multiply (param1 integer,
       param2 integer, out result integer)
   language java parameter style java
   external name 'MathProc.multiply'
```

Использование

- В операторе create procedure можно указать до 31 параметра in, inout и out.
- Чтобы обеспечить совместимость со стандартом ANSI, имена параметров нельзя начинать с символа @. Однако при вызове хранимой процедуры SQLJ из isql или другого клиента, не имеющего отношения к Java, необходимо указать знак @ перед именем параметра, так как это сохраняет порядок именования.
- Синтаксис оператора create procedure в SQLJ отличается от синтаксиса этого оператора в Transact-SQL, чтобы обеспечить совместимость со стандартом SQLJ ANSI. Хранимые процедуры, относящиеся к одному и тому же типу, выполняются одинаковым образом.

Полномочия

По умолчанию полномочия на выполнение команды create procedure принадлежат владельцу базы данных, который может передать их другим пользователям. Полномочия на использование процедуры должны явно предоставляться командой grant и могут отзываться командой revoke.

См. также

Команды create function (SQLJ), drop procedure

Системные процедуры sp_depends, sp_help, sp_helpjava, sp_helprotect

create proxy_table

Описание

Используется *только со службами Component Integration Services*. Создает прокси-таблицу без указания списка столбцов. Службы Component Integration Services получают список столбцов из метаданных удаленной таблицы.

Синтаксис

create proxy_table *имя_таблицы*
[on *имя_сегмента*]
[external [table | directory | file]]
at *путь*

Параметры

имя_таблицы

Локальное имя прокси-таблицы, которое будет использоваться последующими операторами. *имя_таблицы* имеет формат *имя_базы_данных.владелец.объект*, где *имя_базы_данных* и *владелец* являются необязательными и представляют локальные имена базы данных и владельца. Если *имя_базы_данных* не указано, таблица создается в текущей базе данных; если не указан *владелец*, владельцем таблицы становится текущий пользователь. Если указан любой из параметров *имя_базы_данных* или *владелец*, то *имя_таблицы* необходимо заключить в кавычки. Если задан только параметр *имя_базы_данных*, вместо параметра *владелец* необходимо указать метку-заполнитель.

on имя_сегмента

Сегмент, содержащий удаленную таблицу.

external table

Определяет, является ли объект удаленной таблицей или представлением. Установка по умолчанию – external table, так что эта инструкция является необязательной.

external directory

Указывает, что объект – каталог, путь к которому задается в следующем формате: "*/tmp/имя_каталога* [;R]". "R" обозначает "рекурсивный" ("recursive").

external file

Указывает, что объект – файл, путь к которому задается в следующем формате: "*/tmp/имя_файла*".

at путь

Определяет местоположение удаленного объекта. Параметр *путь* имеет формат *имя_сервера.имя_базы_данных.владелец.объект*, где:

- *имя_сервера* (указывать обязательно) – имя сервера, содержащего удаленный объект.

- *имя_базы_данных* (указывать необязательно) – имя базы данных, управляемой удаленным сервером, содержащим удаленный объект.
- *владелец* (указывать необязательно) – имя пользователя удаленного сервера, владеющего удаленным объектом.
- *объект* (указывать обязательно) – имя удаленной таблицы или представления.

Примеры

Создание прокси-таблицы t1, связанной с удаленной таблицей t1. Службы Component Integration Services извлекают список столбцов из удаленной таблицы:

```
create proxy_table t1
at "SERVER_A.db1.joe.t1"
```

Использование

- Команда create proxy_table является вариантом команды create existing table. Команда create proxy_table используется для создания прокси-таблиц, но (в отличие от команды create existing table) не требует указывать список столбцов. Службы Component Integration Services получают список столбцов из метаданных удаленной таблицы.
- В инструкции at задается та же информация о расположении, что и в системной процедуре sp_addobjectdef. Эти сведения хранятся в таблице sysattributes.
- Если объект удаленного сервера не существует, команда возвращает сообщение об ошибке.
- Если объект существует, локальные системные таблицы обновляются. При этом используются все столбцы и извлекаются столбцы таблицы или представления и этих столбцов.
- Службы Component Integration Services автоматически преобразуют тип данных столбца в тип данных Adaptive Server. Если преобразование невозможно, команда create proxy_table не позволяет определить таблицу.
- Из удаленной таблицы извлекается информация об индексах, которая необходима для создания строк в системной таблице sysindexes. Это позволяет определить индексы и ключи в терминах Adaptive Server, благодаря чему оптимизатор запросов может учитывать все индексы, которые могли быть построены по этой таблице.
- Определив прокси-таблицу, вызовите для нее команду update statistics. Это позволит оптимизатору запросов делать разумный выбор порядка соединения.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на выполнение команды create proxy_table принадлежат владельцу таблицы и не могут быть переданы другим пользователям.

См. также

Команды create existing table, create table

create role

Описание

Создает пользовательскую роль; задает срок действия пароля, минимальную длину пароля и максимальное количество неудачных попыток входа в систему, разрешенных для указанной роли.

Синтаксис

create role *имя_роли* [with passwd "*пароль*"
[, {"passwd expiration" | "min passwd length" |
"max failed_logins" } *значение_параметра*]]

Параметры

имя_роли
Имя новой роли. Должно быть допустимым идентификатором, уникальным в пределах сервера. Не может быть переменной.

with passwd
Присоединяет пароль, который пользователь должен ввести, чтобы активировать роль.

пароль
Пароль, который присоединяется к роли. Пароли должны быть допустимыми идентификаторами длиной не менее шести символов. В качестве паролей нельзя использовать переменные.

passwd expiration
Определяет срок действия пароля в днях. Может принимать любое значение от 0 до 32767 включительно.

min passwd length
Определяет минимальную длину пароля для указанной роли.

max failed_logins
Определяет максимальное количество неудачных попыток входа в систему для указанного регистрационного имени.

значение_параметра
Задает значение параметров passwd expiration, min passwd length или max failed_logins.

Примеры

Пример 1. Создание роли doctor_role:

```
create role doctor_role
```

Пример 2. Создание роли doctor_role с паролем physician:

```
create role doctor_role with passwd "physician"
```

Пример 3. Установка срок действия пароля для роли intern_role:

```
create role intern_role, with passwd "temp244",
passwd expiration 7
```

Пример 4. Установка максимального количества неудачных попыток входа в систему для роли intern_role:

```
create role intern_role with passwd “temp244”,
max failed_logins 20
```

Пример 5. Установка минимальной длины пароля для роли intern_role:

```
create role intern_role with passwd "temp244",
min passwd length 0
```

Использование

- Команда create role создает роль с указанными привилегиями, полномочиями и ограничениями. Дополнительную информацию о применении команды create role см. в книге *Руководство по системному администрированию*.

  О мониторинге и ограничении доступа к объектам см. описание команды set role.

- Команду create role нужно выполнять из базы данных master.

- Для присоединения пароля при создании роли нужно указать инструкцию with passwd пароль. Если к роли присоединен пароль, пользователь, которому предоставлена эта роль, должен ввести пароль, чтобы ее активировать.

  О добавлении пароля к роли после ее создания см. в описании команды alter role.

  Примечание. Срок действия паролей, присоединенных к пользовательским ролям, не ограничен.

- Имена ролей должны быть уникальными в пределах сервера.

- Имена ролей не могут совпадать с именами пользователей. Вы можете создать роль, имя которой совпадает с именем пользователя, но при предоставлении привилегий Adaptive Server разрешает конфликт имен, предоставляя привилегии пользователю, а не роли.

  Подробности о конфликтах именования см. в описании команды grant role.

Ограничения

- Во время сеанса сервера можно создать до 1024 ролей. Однако 32 из них зарезервированы под системные роли Sybase, например sa_role и sso_role. Таким образом, максимальное количество пользовательских ролей, которые могут быть созданы во время сеанса сервера, равно 992.

- Если роль создана с присоединенным к ней паролем, пользователь не может активировать ее по умолчанию при входе в систему. Не создавайте роли с паролем, если пользователю нужно по умолчанию активировать роль при входе в систему.

Стандарты

Уровень соответствия стандарту SQL92: расширение языка Transact-SQL.

Полномочия

Команду create role может выполнять только администратор по безопасности.

Полномочия на выполнение команды create role не входят в список полномочий, предоставляемых командой grant all.

См. также

Команды alter role, drop role, grant, revoke, set

Системные процедуры sp_activeroles, sp_displaylogin, sp_displayroles, sp_helprotect, sp_modifylogin

create rule

Описание

Определяет область допустимых значений для определенного столбца или любого столбца пользовательского типа данных и создает правила доступа.

Синтаксис

```
create [ [ and | or ] access]] rule
        [владелец.]имя_правила
        as условное_выражение
```

Параметры

access

Указывает, что создается правило доступа. О правилах доступа см. главу 11 “Управление пользовательскими полномочиями” книги *Руководство по системному администрированию*.

имя_правила

Имя нового правила. Должно быть допустимым идентификатором, не может быть переменной. Чтобы создать правило с тем же именем, что и существующее правило, принадлежащее другому пользователю в текущей базе данных, перед именем правила нужно указать имя владельца. По умолчанию *владелец* – текущий пользователь.

условное_выражение

Задает условия, определяющие правило. Представляет собой выражение, допустимое для инструкции where, может содержать арифметические операторы, операторы отношения, ключевые слова in, like, between и т.п. Это выражение не может содержать имя столбца или другого объекта базы данных, но *может* содержать встроенные функции, не ссылающиеся на объекты базы данных.

Параметр *условное_выражение* содержит единственный аргумент. Аргумент начинается с префикса “@” и ссылается на значение, вставляемое в столбец командами update или insert. При написании правила можно использовать любые имена и символы для обозначения аргумента, но первым символом обязательно должен быть знак “@”. Символьные константы и даты нужно заключать в кавычки, а для двоичных констант использовать префикс “0x”.

Примеры

Пример 1. Создание правила limit, указывающего, что значения столбца advance не могут превышать $1000:

```
create rule limit
as @advance < $1000
```

Пример 2. Создание правила pubid_rule, указывающего, что столбец pub_id может быть равен только 1389, 0736 или 0877:

```
create rule pubid_rule
```

```
as @pub_id in ('1389', '0736', '0877')
```

Пример 3. Создание правила picture, указывающего, что значения столбца value должны начинаться с указанных в правиле символов:

```
create rule picture
as @value like '_-%[0-9]'
```

Использование

- Текст правила можно скрыть с помощью системной процедуры sp_hidetext.
- Правило можно переименовать с помощью процедуры sp_rename.

Ограничения

- Правило можно создать только в текущей базе данных.
- Правило не распространяется на данные, которые уже хранятся в базе данных в момент его создания.
- Операторы create rule нельзя объединять с другими операторами в один пакет.
- Правило нельзя назначать стандартным типам данных Adaptive Server и столбцам типа text, image и timestamp.
- Прежде чем создать новое правило с тем же именем, что и существующее, необходимо удалить правило, а перед удалением правила нужно отменить его привязку к столбцам и типам данных. Для отмены привязки используется следующая процедура:

  ```
  sp_unbindrule objname [, futureonly]
  ```

Назначение правил столбцам и типам данных

- Правило можно назначить столбцу или пользовательскому типу данных с помощью процедуры sp_bindrule. Ее синтаксис следующий:

  ```
  sp_bindrule имя_правила, имя_объекта [, futureonly]
  ```

- Правило, назначенное пользовательскому типу данных, активируется при вставке или обновлении столбца этого типа. Правила *не* проверяют значения, присваиваемые переменным пользовательского типа.
- Правило должно быть совместимо с типом данных столбца. Например, нельзя использовать правило

  ```
  @value like A%
  ```

  для столбца числового типа данных с точным или приближенным представлением чисел. Если правило несовместимо со столбцом, которому оно назначено, Adaptive Server генерирует сообщение об ошибке при попытке вставить значение в этот столбец (а не при назначении правила).

- Правило можно назначить столбцу или типу данных, не отменяя существующего правила.
- Правила, назначенные столбцу, всегда имеют приоритет над правилами, назначенными пользовательским типам данных, независимо от того, какое из них было назначено последним. В таблице 7-11 показано, что будет при назначении правил столбцу и пользовательскому типу данных, у которых уже есть правила.

Таблица 7-11. Приоритет назначения правил

Объект, которому назначено новое правило	Старое правило назначено пользовательскому типу данных	Старое правило назначено столбцу
Пользовательский тип данных	Новое правило заменяет старое	Без изменения
Столбец	Новое правило заменяет старое	Новое правило заменяет старое

Правила и значения NULL

- Правила не отменяют настроек, указанных в определении столбцов. Если правило привязано к столбцу, допускающему значения NULL, то значение NULL можно явно или неявно вставить в столбец, даже если оно не содержится в тексте правила. Например, если столбцу, допускающему значения NULL, назначить правило “@val in (1,2,3)” или “@amount > 10000”, в этот столбец по-прежнему можно вставлять значения NULL. Таким образом, определение столбца имеет приоритет над правилом.

Правила и значения по умолчанию

- Если столбцу назначено как правило, так и значение по умолчанию, то значение по умолчанию должно удовлетворять правилу. Значение по умолчанию, противоречащее правилу, никогда не будет вставлено. При попытке вставить такое значение Adaptive Server генерирует сообщение об ошибке.

Использование ограничений целостности вместо правил

- Правила можно определять с помощью инструкции check оператора create table, которая создает ограничение целостности. Однако такие ограничения действуют для одной таблицы, и их нельзя назначить другим таблицам. Об ограничениях целостности см. в описании операторов create table и alter table.

Получение информации о правилах

- Сведения о правиле можно получить с помощью системной процедуры sp_help.

- Для отображения текста правила, который хранится в системной таблице syscomments, выполните системную процедуру sp_helptext, указав имя нужного правила в качестве параметра.
- После того как правило было назначено столбцу или пользовательскому типу данных, идентификатор этого столбца или типа записывается в системную таблицу syscolumns или systypes соответственно.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Для создания правил с использованием синтаксиса, совместимого с SQL92, используйте инструкцию check оператора create table.

Полномочия

По умолчанию полномочия на выполнение команды create rule принадлежат владельцу базы данных, который может передать их другим пользователям.

См. также

Команды alter table, create default, create table, drop rule, drop table

Системные процедуры sp_bindrule, sp_help, sp_helptext, sp_hidetext, sp_rename, sp_unbindrule

create schema

Описание

Создает новую коллекцию таблиц, представлений и полномочий для пользователя базы данных.

Синтаксис

create schema authorization *имя_авторизации*
оператор_создания_объекта
[оператор_создания_объекта...]
[*оператор_назначающий_полномочия* ...]

Параметры

имя_авторизации
Имя текущего пользователя базы данных.

оператор_создания_объекта
Оператор create table или create view.

оператор_назначающий_полномочия
Оператор grant или revoke.

Примеры

Создание таблиц newtitles, newauthors, newtitleauthors, представления tit_auth_view и соответствующих полномочий:

```
create schema authorization pogo
    create table newtitles (
        title_id tid not null,
        title varchar(30) not null)

    create table newauthors (
        au_id id not null,
        au_lname varchar(40) not null,
        au_fname varchar(20) not null)

    create table newtitleauthors (
        au_id id not null,
        title_id tid not null)

    create view tit_auth_view
    as
        select au_lname, au_fname
            from newtitles, newauthors,
                newtitleauthors
        where
        newtitleauthors.au_id = newauthors.au_id
        and
        newtitleauthors.title_id =
             newtitles.title_id

    grant select on tit_auth_view to public
    revoke select on tit_auth_view from churchy
```

Использование

- Схемы можно создать только в текущей базе данных.

- *имя_авторизации*, называемое также **авторизационным идентификатором схемы**, должно соответствовать имени текущего пользователя.
- Пользователь должен обладать соответствующими полномочиями на выполнение команд (create table и/или create view). Если пользователь создает представление по таблице, принадлежащей другому пользователю, полномочия проверяются не при создании представления, а при попытке доступа к данным через это представление.
- Команда create schema завершается:
  - стандартными символами завершения команды (в isql по умолчанию используется “go”);
  - любым оператором, кроме create table, create view, grant и revoke.
- Если любая команда в операторе create schema завершается ошибкой, происходит откат всего оператора и ни одна из его команд не выполняется.
- Оператор create schema добавляет информацию о таблицах, представлениях и полномочиях в системные таблицы. Для удаления объектов, созданных оператором create schema, используйте соответствующие команды удаления (drop table или drop view). Набор полномочий, предоставленных пользователю или отозванных у него через схему, можно изменить стандартными командами grant и revoke вне оператора создания схемы.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду create schema может выполнять любой пользователь базы данных. Пользователь должен иметь полномочия на создание объектов, указанных в схеме (то есть полномочия create table и/или create view).

См. также

Команды create table, create view, grant, revoke

Утилиты isql

create table

Описание

Создает новую таблицу, а также может создавать ограничения целостности.

Синтаксис

```
create table [база_данных .[владелец ].]имя_таблицы (имя_столбца
тип_данных
        [default {константа | user | null}]
        {[{identity | null | not null}]
            [off row | [ in row [ (размер_в_байтах) ] ]
        [[constraint имя_ограничения ]
            {{unique | primary key}
            [clustered | nonclustered] [asc | desc]
            [with { fillfactor = pct,
                    max_rows_per_page = количество_строк, }
                    reservepagegap = количество_страниц }]
            [ on имя_сегмента]
            | references [[база_данных .]владелец .]родительская_таблица
                [(родительский_столбец )]
            | check (условие_поиска)}]}...
        | [constraint имя_ограничения]
            {{unique | primary key}
                [clustered | nonclustered]
                (имя_столбца [asc | desc]
                    [{, имя_столбца [asc | desc]}...])
                [with { fillfactor = pct
                    max_rows_per_page = количество_строк,
                    reservepagegap = количество_страниц } ]
                [ on имя_сегмента]
            |foreign key (имя_столбца [{,имя_столбца}...])
            | references [[база_данных.]владелец.]родительская_таблица
                    [(родительский_столбец [{, родительский_столбец}...])]
            | check (условие_поиска) ... }
        [{, {следующий_столбец | следующее_ограничение}}...])
        [lock {datarows | datapages | allpages }]
        [with { max_rows_per_page = количество_строк,
            exp_row_size = количество_байтов,
            reservepagegap = количество_страниц,
            identity_gap = значение } ]
        [ on имя_сегмента ]
        [ [ external table ] at имя_пути ]
```

Параметры

имя_таблицы

Явное имя новой таблицы. Имя таблицы должно содержать имя базы данных, если она находится в другой базе данных, и имя владельца, если в базе данных существует несколько таблиц с таким именем. По умолчанию параметры *владелец* и *база_данных* равны соответственно текущему пользователю и текущей базе данных.

Для указания имени таблицы нельзя использовать переменную. Это имя должно быть уникальным в базе данных и среди всех объектов, принадлежащих данному владельцу. Если параметр set quoted_identifier

установлен в on, то в имени таблицы можно использовать разделители. В противном случае имя таблицы должно быть допустимым идентификатором. Дополнительную информацию о допустимых именах таблиц см. в разделе "Идентификаторы" главы 4, "Выражения, идентификаторы и метасимволы."

Чтобы создать временную таблицу, перед именем таблицы нужно указать символ # или префикс "tempdb..". Дополнительную информацию см. в разделе "Таблицы, начинающиеся с # (временные таблицы)" главы 4, "Выражения, идентификаторы и метасимволы."

Пользователь может создать таблицу в другой базе данных, если он содержится в таблице sysusers и обладает полномочиями create table в этой базе данных. Каждый из перечисленных ниже операторов создает таблицу newtable в базе данных otherdb:

```
create table otherdb..newtable
create table otherdb.yourname.newtable
```

имя_столбца

Имя столбца в таблице. Должно быть уникальным в таблице. Команда set quoted_identifier on позволяет использовать разделители в имени столбца. В противном случае имя столбца должно быть допустимым идентификатором. Дополнительную информацию о допустимых именах столбцов см. в главе 4, "Выражения, идентификаторы и метасимволы."

тип_данных

Тип данных столбца. Может быть системным или пользовательским. Для некоторых типов данных указывается длина (*n*) в круглых скобках:

```
тип_данных(n)
```

Для других – точность (*p*) и масштаб (*s*):

```
тип_данных(p,s)
```

Дополнительную информацию см. в разделе "Типы данных".

Если в базе данных включена поддержка Java, *тип_данных* может быть системным или пользовательским классом Java, установленным в базе данных. Дополнительную информацию см. в книге *Java in Adaptive Server Enterprise*.

default

Указывает значение по умолчанию для столбца. Если задано значение по умолчанию и пользователь не указывает значение столбца при вставке данных, Adaptive Server вставляет значение по умолчанию.

В качестве значений по умолчанию можно использовать постоянное выражение user для вставки имени пользователя, выполнившего операцию, или null для вставки значения NULL. Adaptive Server создает имя значения по умолчанию в формате *таблица_столбец_идентификатор_объекта*, где *таблица* – первые 10 символов имени таблицы, *столбец* – первые 5 символов имени столбца и *идентификатор_объекта* – идентификатор значения по умолчанию. Значения по умолчанию, объявленные для столбца IDENTITY, не влияют на его значения.

константа

Значение по умолчанию для столбца. Не может содержать глобальные переменные, имена столбцов и другие объекты базы данных, но может содержать встроенные функции, не ссылающиеся на объекты базы данных. Если значение по умолчанию несовместимо с типом данных столбца, то при попытке вставить такое значение в столбец Adaptive Server сгенерирует ошибку.

user | null

Указывает, что Adaptive Server должен вставить имя пользователя или NULL, если пользователь не укажет значение. Ключевое слово user можно указать для столбцов типа данных char(30) или varchar(30). Ключевое слово null можно указать для столбцов, допускающих значения NULL.

identity

Указывает, что указанный столбец обладает свойством IDENTITY. Каждая таблица в базе данных может содержать один столбец IDENTITY, имеющий тип numeric с масштабом 0. Столбцы IDENTITY не могут быть обновлены и не допускают значений NULL.

Столбцы IDENTITY используются для хранения последовательных номеров, например номеров счетов или номеров служащих, которые автоматически генерируются Adaptive Server. Значение столбца IDENTITY однозначно идентифицирует каждую строку таблицы.

null | not null

Определяет действия сервера Adaptive Server при вставке данных в столбец, если значение по умолчанию не указано.

Ключевое слово null указывает, что Adaptive Server вставит в столбец значение NULL, если пользователь не укажет значение для этого столбца.

Ключевое слово not null указывает, что пользователь должен предоставить значение, отличное от NULL, если значение по умолчанию отсутствует.

Если ключевые слова null или not null не указаны, по умолчанию принимается not null. Однако эту установку по умолчанию можно изменить процедурой sp_dboption, чтобы значение по умолчанию было совместимо со стандартами SQL.

off row | in row
: Указывает, будут ли значения столбца Java-SQL храниться отдельно от строки (off row) или в области, выделенной для строки (in row).

  Установка по умолчанию – off row. Дополнительную информацию об этом см. в книге *Java in Adaptive Server Enterprise*.

размер_в_байтах
: Определяет максимальный размер столбца, хранящегося в строке (то есть с установкой “in-row”).Объект, хранящийся в строке, может занимать до 16 КБ (точное значение зависит от размера страницы сервера базы данных и других переменных).

constraint
: Вводит имя ограничения целостности.

имя_ограничения
: Имя ограничения. Должно быть допустимым идентификатором и уникальным в базе данных. Если имя ограничения ссылочной целостности или ограничения check не указано, Adaptive Server создает имя в формате *таблица_столбец_идентификатор_объекта*, где *таблица* – первые 10 символов имени таблицы, *столбец* – первые 5 символов имени столбца и *идентификатор_объекта* – объектный идентификатор ограничения. Если не указано имя ограничения unique или ограничения первичного ключа, Adaptive Server создает имя в формате *таблица_столбец_идентификатор_табл/инд*, где *идентификатор_табл/инд* – строка, полученная путем конкатенации идентификаторов таблицы и индекса.

unique
: Запрещает одинаковые значения в столбце или столбцах. Это ограничение создает уникальный индекс, который удаляется только при удалении ограничения командой alter table.

primary key
: Запрещает одинаковые значения в столбце или столбцах и значения NULL. Это ограничение создает уникальный индекс, который удаляется только при удалении ограничения командой alter table.

clustered | nonclustered
: Указывает, является ли индекс, созданный ограничениями unique или primary key, кластерным или некластерным. Для ограничения первичного ключа по умолчанию создается кластерный индекс, а для огра-

ничения unique – некластерный. У таблицы может быть только один кластерный индекс. Дополнительную информацию см. в описании команды create index.

asc | desc

Определяет порядок сортировки столбцов (по возрастанию или по убыванию) для индекса, созданного ограничением. По умолчанию данные сортируются по возрастанию.

fillfactor

Указывает, на сколько процентов заполняются страницы при создании нового индекса по существующим данным. Установка fillfactor применяется только при создании индекса. При обновлении данных процент заполнения страниц меняется.

По умолчанию значение fillfactor равно 0; оно используется, если в операторе create index не указано ключевое слово with fillfactor (если значение по умолчанию не было изменено процедурой sp_configure). Параметр fillfactor может принимать значения от 1 до 100.

Если параметр fillfactor равен 0, все страницы кластерного индекса и листовые страницы некластерного индекса заполняются на 100 процентов. При этом B-дереве как кластерного, так и некластерного индекса остается достаточно места. Чаще всего нет необходимости менять параметр fillfactor.

Если параметр fillfactor равен 100, то как для кластерного, так и для некластерного индекса все страницы будут заполнены на 100 процентов. Такое значение параметра fillfactor имеет смысл только для таблиц, предназначенных только для чтения, то есть таблиц, в которые никогда не будут добавлены данные.

Если значение fillfactor меньше 100 (за исключением 0, являющегося особым случаем), в страницах создаваемых впоследствии индексов будет оставаться свободное место. Для fillfactor рекомендуется задать значение 10, если таблица, для которой создается индекс, впоследствии будет содержать значительно больше данных, чем в текущий момент. Однако необходимо иметь в виду, что чем меньше значение fillfactor, тем больше пространства занимает индекс (или индекс и данные).

Параметр fillfactor нельзя использовать для удаленных серверов, если включены службы Component Integration Services.

Предупреждение. Значение параметра fillfactor влияет на размер пространства, которое занимают данные, так как Adaptive Server перераспределяет их при создании кластерного индекса.

---

max_rows_per_page

Ограничивает количество строк на страницах данных и листовых страницах индексов. В отличие от fillfactor, параметр max_rows_per_page будет ограничивать количество строк на странице при вставке и удалении данных.

Если значение max_rows_per_page не указано, при создании таблицы используется значение 0. Для таблиц и кластерных индексов параметр может принимать значение от 0 до 256. Максимальное количество строк на странице некластерного индекса зависит от размера ключа индекса, Adaptive Server возвращает сообщение об ошибке, если указанное значение превышает максимальную величину.

Если параметр max_rows_per_page равен 0, все страницы данных кластерного индекса и листовые страницы некластерного индекса заполняются полностью. При этом B-дереве как кластерного, так и некластерного индекса остается достаточно пространства.

Небольшие значения параметра max_rows_per_page позволяют снизить конкуренцию между блокировками за часто используемые данные. Однако в этом случае страницы создаваемого индекса будут заняты не полностью, в результате чего увеличивается объем занимаемого пространства и вероятность расщепления страниц.

Если службы Component Integration Services включены и создается прокси-таблица, то параметр max_rows_per_page игнорируется. Прокси-таблицы не содержат данных. Если таблица создана с параметром max_rows_per_page, а затем создается прокси-таблица, ссылающаяся на эту таблицу, то ограничения, заданные параметром max_rows_per_page, применяются при вставке и удалении через прокси-таблицу.

on имя_сегмента

Указывает, что индекс должен быть создан в заданном сегменте. Перед использованием параметра on имя_сегмента необходимо инициализировать устройство с помощью команды disk init и добавить сегмент к базе данных с помощью системной процедуры sp_addsegment. Список сегментов базы данных можно узнать у системного администратора или с помощью процедуры sp_helpsegment.

Если одновременно указаны параметры clustered и on имя_сегмента, вся таблица перемещается в указанный сегмент, так как страницы с данными хранятся на уровне листа индекса.

references

Определяет список столбцов для ограничения ссылочной целостности. Для ограничения, определяемого на уровне столбца, можно указать только один столбец. Если это ограничение включено в таблицу, ссы-

лающуюся на другую таблицу, то все данные, которые вставляются в *дочернюю* (ссылающуюся) таблицу, должны уже существовать в *родительской* таблице (таблице, на которую ссылается дочерняя таблица).

Для использования этого ограничения необходимо обладать полномочиями references для родительской таблицы. Указанные столбцы в родительской таблице должны быть ограничены уникальным индексом (созданным ограничением unique или оператором create index). Если столбцы не указаны, то в родительской таблице должно быть определено ограничение primary key на соответствующих столбцах. Кроме того, типы данных соответствующих столбцов дочерней и родительской таблиц должны попарно совпадать.

foreign key
Определяет список столбцов, составляющих внешние ключи этой таблицы. Эти ключи ссылаются на столбцы, перечисленные в инструкции references, идущей за этой инструкцией. Этот синтаксис допустим только для ограничения уровня таблицы, но не для ограничения уровня столбца.

родительская_таблица
Имя родительской таблицы. Родительская таблица может находиться в другой базе данных. Ограничения могут ссылаться не более чем на 192 пользовательские таблицы и рабочие таблицы, созданные системой.

родительский_столбец
Имя столбца или столбцов в родительской таблице.

check
Указывает *условие_поиска*, которое будет проверяться для всех строк в таблице. Ограничение check может распространяться на таблицу или столбец; определение столбца в операторе create table может содержать несколько ограничений check.

условие_поиска
Ограничение check для значений столбцов. Оно может содержать:

- Список постоянных выражений, предваряемый ключевым словом in.
- Предваряемый ключевым словом like набор условий, которые могут содержать метасимволы.

Ограничения check, определенные на уровне столбца и таблицы, могут ссылаться на любой столбец таблицы.

Выражение может содержать арифметические операции и функции, но *условие_поиска* не может содержать подзапросы, агрегатные функции, переменные базового языка и параметры.

следующий_столбец | *следующее_ограничение*
Означает, что можно включать дополнительные определения столбцов и ограничений уровня таблицы (разделенных запятыми), используя описанный выше синтаксис.

lock datarows | datapages | allpages
Определяет схему блокировки таблицы. По умолчанию используется значение параметра конфигурации сервера lock scheme.

exp_row_size = количество_байтов
Задает ожидаемый размер строки; применяется только в схемах блокировки datarows (блокировка строк данных) и datapages (блокировка страниц данных) и только для таблиц со строками переменной длины. Допустимыми значениями являются 0, 1 и другие значения между минимальной и максимальной длиной строки в таблице. По умолчанию используется значение 0, которое означает, что применяется установка, заданная на уровне сервера.

reservepagegap = количество_страниц
Указывает отношение количества заполненных страниц к количеству страниц, которые необходимо оставить пустыми для будущего выделения буферов ввода/вывода экстентов. Для будущего расширения таблицы будет оставлено по одной пустой странице на заданное *количество_страниц*. Допустимые значения – от 0 до 255. Значение по умолчанию равно 0.

with identity_gap
Определяет интервал между значениями типа identity для таблицы. Это значение имеет приоритет над значением соответствующего системного параметра, но только для данной таблицы.

значение
Величина интервала между значениями типа identity. Дополнительную информацию об установке интервала между значениями типа identity см. в разделе "Столбцы IDENTITY".

external table
Указывает, что объект является удаленной таблицей или представлением. Параметр external table используется по умолчанию, поэтому эта инструкция является необязательной.

at имя_пути

Указывает местоположение удаленного объекта. Параметр *имя_пути* имеет формат *имя_сервера.имя_базы_данных.владелец;дополнение1.дополнение2*, где:

- *имя_сервера* (указывать обязательно) – имя сервера, содержащего удаленный объект;
- *имя_базы_данных* (указывать необязательно) – имя базы данных, управляемой удаленным сервером, содержащим удаленный объект;
- *владелец* (указывать необязательно) – имя пользователя удаленного сервера, владеющего удаленным объектом;
- *объект* (указывать обязательно) – имя удаленной таблицы или представления.
- *дополнение1.дополнение2*(указывать необязательно) – строковое выражение, передаваемое серверу во время выполнения команд create table или create index. Эта строка используется только для серверов класса db2. Дополнение1 – база данных DB2, где нужно разместить таблицу, и дополнение2 – табличное пространство DB2, где нужно разместить таблицу.

on имя_сегмента

Имя сегмента, где нужно разместить таблицу. Чтобы можно было указать инструкцию on имя_сегмента, логическое устройство должно быть назначено базе данных командой create database или alter database, а сегмент должен быть добавлен в базу данных процедурой sp_addsegment. Список сегментов базы данных можно узнать у системного администратора или с помощью процедуры sp_helpsegment.

Примеры

Пример 1. Создание таблицы titles:

```
create table titles
(title_id tid not null,
title varchar(80) not null,
type char(12) not null,
pub_id char(4) null,
price money null,
advance money null,
total_sales int null,
notes varchar(200) null,
pubdate datetime not null,
contract bit not null)
```

Пример 2. Создание таблицы compute. Имя таблицы и имена столбцов max и min заключены в двойные кавычки, так как являются зарезервированными словами. Имя столбца total score заключено в двойные кавычки, так как оно содержит пробел. Перед созданием таблицы необходимо выполнить команду set quoted_identifier on:

```
create table "compute"
("max" int, "min" int, "total score" int)
```

Пример 3. Одновременное создание таблицы sales и кластерного индекса с ограничением unique (в сценарии установки базы данных pubs2 таблица и индекс создаются отдельно друг от друга операторами create table и create index):

```
create table sales
(stor_id         char(4)      not null,
ord_num          varchar(20)  not null,
date             datetime     not null,
unique clustered (stor_id, ord_num))
```

Пример 4. Создание таблицы salesdetail с двумя ограничениями ссылочной целостности и одним значением по умолчанию. Ограничение ссылочной целостности salesdet_constr определяется на уровне таблицы, а ограничение без имени – на уровне столбца title_id. Оба ограничения указывают столбцы, по которым построены уникальные индексы в родительских таблицах (titles и sales). Инструкция default определяет значение по умолчанию равное 0 для столбца qty:

```
create table salesdetail
(stor_id    char(4)                 not null,
ord_num     varchar(20)             not null,
title_id     tid                    not null
             references titles(title_id),
qty          smallint default 0     not null,
discount    float                   not null,

constraint salesdet_constr
    foreign key (stor_id, ord_num)
    references sales(stor_id, ord_num))
```

Пример 5. Создание таблицы publishers с проверочным ограничением check для столбца pub_id. Это ограничение уровня столбца используется вместо правила pub_idrule базы данных pubs2:

```
create rule pub_idrule
as @pub_id in ("1389", "0736", "0877", "1622",
    "1756")
or @pub_id like "99[0-9][0-9]"

create table publishers
```

```
(pub_id char(4) not null
    check (pub_id in ("1389", "0736", "0877", "1622",
        "1756")
    or pub_id like "99[0-9][0-9]"),
pub_name     varchar(40)  null,
city     varchar(20)  null,
state     char(2)  null)
```

Пример 6. Создание таблицы sales_daily со столбцом identity ord_num. Когда в таблицу в первый раз вставляется строка, СУБД Adaptive Server присваивает столбцу identity значение 1. При каждой следующей вставке значение в этом столбце увеличивается на 1:

```
create table sales_daily
(stor_id         char(4)         not null,
ord_num          numeric(10,0)   identity,
ord_amt          money           null)
```

Пример 7. Создание таблицы new_titles со схемой блокировки datapages и ожидаемым размером строки, равным 200:

```
create table new_titles (
      title_id    tid,
      title       varchar(80) not null,
      type        char(12) ,
      pub_id      char(4) null,
      price       money null,
      advance     money null,
      total_sales int null,
      notes       varchar(200) null,
      pubdate     datetime,
      contract    bit                          )
lock datapages
with exp_row_size = 200
```

Пример 8. Создание таблицы со схемой блокировки datarows и параметром reservepagegap, равным 16 (это означает, что операции ввода-вывода при выделении экстента оставляют одну свободную страницу на 15 заполненных страниц):

```
create table new_publishers (
pub_id          char(4) not null,
pub_name        varchar(40) null,
city            varchar(20) null,
state           char(2) null )
lock datarows
with reservepagegap = 16
```

Пример 9. Создание ограничения, основанного на уникальном кластерном индексе. В этом индексе значения столбца stor_id упорядочены по возрастанию, а по значения столбца ord_num – по убыванию:

```
create table sales_south
(stor_id         char(4)      not null,
ord_num          varchar(20)  not null,
date             datetime     not null,
unique clustered (stor_id asc, ord_num desc))
```

Пример 10. Создание таблицы t1 на удаленном сервере SERVER_A и прокси-таблицы t1, сопоставленной с удаленной таблицей:

```
create table t1
(a        int,
 b        char(10))
at "SERVER_A.db1.joe.t1"
```

Пример 11. Создание таблицы employees со столбцом name типа varchar и столбцами Java-SQL home_addr типа Address и mailing_addr типа Address2Line. Типы Address и Address2Line являются классами Java, установленными в базе данных:

```
create table employees
(name varchar(30),
home_addr Address,
mailing_addr Address2Line)
```

Пример 12. Создание таблицы mytable со столбцом identity. Интервал между значениями identity равен 10, то есть идентификаторы выделяются блоками по 10. Если работа сервера будет закончена командой shutdown с параметром nowait или произойдет сбой, то разность между последним идентификатором, присвоенным строке, и следующим идентификатором не превысит десяти:

```
create table mytable
(IdNum numeric(12,0) identity)
with identity_gap = 10
```

Дополнительную информацию об интервалах между значениями identity см. в разделе “Managing Identity Gaps in Tables” главы 7, “Creating Databases and Tables”, книги *Transact-SQL User’s Guide*.

Использование

- Команда create table создает таблицы, а также дополнительно может создавать ограничения целостности. Таблица создается в текущей базе данных, если в этой команде явно не указана другая база данных. Пользователь может создать таблицу или индекс в другой базе данных, если его имя содержится в таблице sysusers и он обладает полномочиями на выполнение команды create table в соответствующей базе данных.

- Пространство под таблицы и индексы выделяется с приращением, равным одному экстенту (восьми страницам). Каждый раз, когда экстент заполняется, выделяется следующий экстент. Размер пространства, выделенного таблице, и пространства, используемого таблицей, можно узнать с помощью системной процедуры sp_spaceused.
- Максимальная длина столбца Java, хранящегося в строке, определяется максимальным размером столбца переменной длины для схемы таблицы, типом блокировки и размером страницы.
- Если оператор create table, создающий таблицу со столбцом char(*n*) NULL, выполняется из служб Component Integration Services, этот столбец создается на удаленном сервере с типом varchar(*n*).

Ограничения

- Максимальное количество столбцов в таблице зависит от ширины столбцов и от размера логической страницы сервера:
  - Сумма размеров столбцов не может превышать размер логической страницы сервера.
  - Таблица может содержать до 1024 столбцов.
  - APL-таблица может содержать до 254 столбцов переменной длины.

    Например, если размер логической страницы сервера равен 2 КБ, то максимальное число целочисленных столбцов в таблице будет гораздо меньше, чем 1024 (1024 * 4 превышает этот предел).

    Таблица может содержать столбцы постоянной и переменной длины, если их количество не превышает 1024. Например, если размер логической страницы сервера равен 8 КБ, APL-таблица может состоять из 254 целочисленных столбцов, допускающих значения NULL (это столбцы переменной длины), и 770 целочисленных столбцов, не допускающих значения NULL. Суммарное количество столбцов в этом случае будет равно 1024.
- База данных может содержать до 2 000 000 000 таблиц, а каждая таблица – до 250 пользовательских столбцов. Количество строк в таблице ограничено только объемом доступной памяти.
- В следующих случаях таблицы будут созданы, но при выполнении последующих операций DML над этими таблицами будет выдано сообщение об ошибке, вызванной нарушением ограничений на размер:

- суммарный размер строк со столбцами переменной длины превышает максимальный размер столбца;
- длина одного столбца переменной длины превышает максимальный размер столбца;
- в DOL-таблице смещение любого столбца переменной длины (кроме начального) превышает ограничение, составляющее 8191 байтов.

- Adaptive Server генерирует ошибку, если суммарный размер всех столбцов постоянной длины и служебной части строки превышает величину, допустимую схемой блокировки и размером страницы. Эти ограничения описаны в таблице 7-12.

Таблица 7-12. Максимальная длина строки и столбца в таблицах APL и DOL

Схема блокировки	Размер страницы	Максимальная длина строки	Максимальная длина столбца
APL-таблицы	2 КБ (2048 байтов)	1962	1960 байтов
	4 КБ (4096 байтов)	4010	4008 байтов
	8 КБ (8192 байта)	8106	8104 байта
	16 КБ (16384 байта)	16298	16296 байтов
DOL-таблицы	2 КБ (2048 байтов)	1964	1958 байтов
	4 КБ (4096 байтов)	4012	4006 байтов
	8 КБ (8192 байта)	8108	8102 байта
	16 КБ (16384 байта)	16300	16294 байта, если таблица не содержит столбцов переменной длины
	16 КБ (16384 байта)	16300 (при условии, что максимальное начальное смещение столбца переменной длины = 8191)	8191-6-2 = 8183 байта, если таблица содержит хотя бы один столбец переменной длины*
	* Включает шесть байтов служебной части строки и два байта поля длины строки.		

- Максимальный размер данных в столбцах переменной длины для одной строки зависит от схемы блокировки таблицы. Максимальные размеры столбцов для APL-таблицы перечислены в таблице 7-13:

Таблица 7-13. Максимальный размер столбцов переменной длины в APL-таблице

Размер страницы	Максимальная длина строки	Максимальная длина столбца
2 КБ (2048 байтов)	1962	1960
4 КБ (4096 байтов)	4010	4008
8 КБ (8192 байта)	8096	8104
16 КБ (16384 байта)	16298	16296

Максимальные размеры столбцов для DOL-таблиц перечислены в таблице 7-14:

Таблица 7-14. Максимальный размер столбцов переменной длины в DOL-таблице

Размер страницы	Максимальная длина строки	Максимальная длина столбца
2 КБ (2048 байтов)	1964	1958
4 КБ (4096 байтов)	4012	4006
8 КБ (8192 байта)	8108	8102
16 КБ (16384 байта)	16300	16294

- Если создать DOL-таблицу со столбцом переменной длины, чье смещение превышает предел, составляющий 8191 байтов, в этот столбец нельзя будет вставлять значения.
- При создании таблицы со столбцами varchar, nvarchar, univarchar или varbinary, суммарная длина которых (указанная при определении) превышает максимально допустимый размер строки, будет выдано предупреждающее сообщение, но таблица будет создана. Если попытаться вставить данные в такую строку или обновить ее оператором update так, чтобы размер строки превысил максимально допустимую длину, команда не будет выполнена и будет выдано сообщение об ошибке.

Примечание. Если оператор create table находится внутри блока if...else или цикла while, схема для таблицы создается перед проверкой истинности условия. Это может привести к ошибке, если таблица уже существует. Убедитесь в том, что таблица с таким именем отсутствует в базе данных.

- В одном пакете или процедуре нельзя сначала выполнить оператор create table, определяющий значение по умолчанию или ограничение check, а затем вставить данные в таблицу. В этом случае нужно либо поместить операторы создания таблицы и вставки в разные пакеты,

либо использовать команду execute, чтобы выполнить эти действия отдельно друг от друга.

- При определении значений по умолчанию в операторе create table нельзя использовать переменные:

```
declare @p int
select @p = 2
create table t1 (c1 int default @p, c2 int)
```

  При выполнении такого оператора появится сообщение об ошибке 154 – “Variable is not allowed in default”.

Определения столбцов

- При создании столбцов типа данных, определенного пользователем:
  - Нельзя изменять длину, точность и масштаб.
  - Тип NULL можно использовать для создания столбцов, не допускающих значения NULL, но не для создания столбцов IDENTITY.
  - Тип NOT NULL можно использовать как для создания столбцов, допускающих значения NULL, так и для столбцов IDENTITY.
  - Для создания столбцов, не допускающих значения NULL, можно использовать тип IDENTITY, но такой столбец унаследует свойство IDENTITY. Тип IDENTITY нельзя использовать для создания столбцов, допускающих значения NULL.
- Значения NULL могут храниться только в столбцах переменной длины. При создании столбца постоянной длины, допускающего значения NULL, Adaptive Server автоматически преобразует его к соответствующему типу данных переменной длины. Adaptive Server не информирует пользователя об изменении типа данных.

  В следующей таблице перечислены типы данных постоянной длины и соответствующие им типы данных переменной длины. Некоторые типы данных переменной длины, например moneyn, зарезервированы и не могут использоваться для создания столбцов, переменных и параметров.

Таблица 7-15. Типы данных переменной длины, используемые для хранения значений NULL

Исходный тип данных постоянной длины	Тип, к которому преобразуется
char	varchar
nchar	nvarchar

Исходный тип данных постоянной длины	Тип, к которому преобразуется
binary	varbinary
datetime	datetimn
float	floatn
int, smallint и tinyint	intn
decimal	decimaln
numeric	numericn
money и smallmoney	moneyn

- Существует два способа создания значений по умолчанию: объявить значение по умолчанию как ограничение уровня столбца в команде create table или alter table или создать значение по умолчанию командой create default и назначить его столбцу процедурой sp_bindefault.
- Для получения информации о таблице и ее столбцах выполните системную процедуру sp_help.

Временные таблицы

- Временные таблицы хранятся во временной базе данных tempdb.
- Первые 13 символов имени временной таблицы должны быть уникальными для сеанса. Доступ к таким таблицам можно получить только в текущем сеансе Adaptive Server. Они хранятся в таблице tempdb..objects (при этом к их имени приписывается справа числовой суффикс, сгенерированный системой) и удаляются после завершения текущего сеанса или при явном удалении таблицы.
- Временные таблицы, созданные с префиксом “tempdb..”, совместно используются пользовательскими сеансами Adaptive Server. Эти таблицы удаляются только явно владельцами или при перезагрузке Adaptive Server. Временные таблицы с префиксом “tempdb..” создаются внутри хранимой процедуры, только в том случае, когда необходимо предоставить совместный доступ пользователям и сеансам. Чтобы избежать ненамеренного предоставления совместного доступа к временным таблицам, при их создании и удалении внутри хранимой процедуры используется префикс “#”.
- Во время сеанса Adaptive Server временные таблицы могут использоваться несколькими пользователями. Однако определить сеанс конкретного пользователя обычно невозможно, так как временные таблицы создаются с “гостевым” идентификатором пользователя, равным 2. Если несколько пользователей запускают процесс, создающий временную таблицу, каждый пользователь является “гостем”, поэтому значения uid совпадают. Таким образом, невозможно узнать, какой сеанс во временной таблице принадлежит конкретному

пользователю. Системный администратор может предоставить пользователю доступ ко временной таблице процедурой sp_addlogin, в этом случае во время сеанса можно использовать индивидуальные uid, но эта ситуация маловероятна.

- Ко временной таблице можно привязать правила, значения по умолчанию и индексы, однако нельзя создать представления и триггеры.
- При создании временной таблицы можно использовать типы данных, определенные пользователем, если они содержатся в таблице tempdb..systypes. Чтобы добавить пользовательский тип данных в базу данных tempdb только для текущего сеанса, выполните процедуру sp_addtype в базе данных tempdb. Чтобы окончательно сохранить тип данных, выполните процедуру sp_addtype в базе данных model и перезапустите Adaptive Server, чтобы база данных model скопировалась в tempdb.

Использование индексов

- Таблица “следует” за кластерным индексом. Если таблица создается на одном сегменте, а кластерный индекс на другом, таблица перемещается на сегмент кластерного индекса.
- Выполнение операций вставки, обновления и выборки данных можно ускорить, поместив таблицу в один сегмент, а некластерные индексы – в другой, если сегменты находятся на разных физических устройствах. Дополнительную информацию см. в книге *Руководство по настройке производительности*.

Переименование таблиц и столбцов

- Для переименования таблиц и столбцов используется процедура sp_rename.
- После переименования таблицы или ее столбцов выясните с помощью процедуры sp_depends, какие процедуры, триггеры и представления зависят от этой таблицы, и переопределите эти объекты.

Предупреждение. Если не переопределить зависимые объекты, они перестанут работать после перекомпиляции.

Определение порядка сортировки в индексах

- Порядок сортировки для индекса задается с помощью ключевых слов asc и desc, которые указываются после имен столбцов индекса. Если в инструкции order by запроса столбцы указаны в том же порядке, в котором они были указаны при создании какого-либо существующего индекса, то при обработке этого запроса можно будет обойтись без сортировки.

Определение ограничений целостности

- Оператор create table помогает управлять целостностью базы данных с помощью ряда ограничений целостности, определенных стандартом SQL. Эти инструкции ограничивают данные, которые пользователь может вставить в столбец. Целостность базы данных обеспечивается также значениями по умолчанию, правилами, индексами и триггерами.

  Ограничения целостности упрощают процесс создания правил целостности, позволяя определить их одной операцией во время создания таблицы. Однако область их применения и возможности ограничены сильнее, чем у значений по умолчанию, правил, индексов и триггеров.

- Ограничения, объявленные на уровне таблицы, затрагивают несколько столбцов, а ограничения, объявленные на уровне столбца, применяются к одному столбцу. Несмотря на то что пользователь редко замечает разницу, ограничения уровня столбца проверяются, только если изменяется значение данного столбца, в то время как ограничения уровня таблицы проверяются при любом изменении строки.

  Ограничения уровня столбца объявляются за именем столбца и типом данных, перед запятой (см. пример 5). Ограничения уровня таблицы объявляются отдельными инструкциями, разделенными запятыми (см. пример 4). Adaptive Server воспринимает ограничения обоих уровней одинаково, поэтому ни одно из них не является более эффективным, чем другое.

- Вы можете создать следующие ограничения на уровне таблицы и столбца:

  - Ограничение unique запрещает повторение значений в указанном столбце таблицы. Кроме того, ограничение primary key запрещает значения NULL в указанном столбце.

  - Ограничение ссылочной целостности (references) указывает, что данные столбца должны соответствовать данным в заданных таблицах и столбцах.

  - Ключевое слово check ограничивает значения, которые можно вставить в указанные столбцы.

  Для обеспечения целостности данных можно также разрешить или запретить значения NULL (ключевые слова null или not null) и определить значения по умолчанию (инструкция default).

- Системные процедуры sp_primarykey, sp_foreignkey и sp_commonkey позволяют сохранить информацию в системных таблицах и сделать понятными отношения между таблицами в базе данных. Эти системные процедуры не приводят в исполнение отношения ключей и не дублируют функции ключевых слов primary key и foreign key в операторе create table. Для просмотра информации о ключах используется процедура sp_helpkey. Для просмотра часто используемых соединений используется процедура sp_helpjoins.
- Transact-SQL предоставляет несколько механизмов обеспечения целостности. Кроме ограничений, объявляемых в операторе create table, можно создавать правила, значения по умолчанию, индексы и триггеры. В таблице 7-16 перечислены все ограничения целостности и описаны другие методы обеспечения целостности:

Таблица 7-16. Обеспечение целостности

При создании таблицы	Другие методы
ограничение unique	команда create unique index (по столбцу, допускающему значения NULL)
ограничение primary key	команда create unique index (по столбцу, не допускающему значения NULL)
ограничение references	create trigger
ограничение check (уровень таблицы)	create trigger
ограничение check (уровень столбца)	операторы create trigger или create rule и процедура sp_bindrule
инструкция default	команда create default и процедура sp_bindefault

Выбор метода зависит от требований. Например, триггеры позволяют управлять более сложными ограничениями ссылочной целостности (включая ссылку на другие столбцы и объекты), чем те, которые можно объявить в операторе create table. Кроме того, ограничения, объявленные в операторе create table, относятся только к одной таблице; в отличие от правил и значений по умолчанию, их нельзя привязать к другим таблицам, а можно только удалить или изменить оператором alter table. Ограничения не могут содержать подзапросы и агрегатные функции.

- Команда create table может включать множество ограничений, удовлетворяющих следующим условиям:
  - Максимальное количество ограничений unique лимитировано количеством индексов, которые можно построить по таблице.

- Таблица может иметь только одно ограничение primary key.
- Для каждого столбца таблицы можно определить только одно значение по умолчанию инструкцией default, но можно создать несколько различных ограничений для одного и того же столбца.

Например:

```
create table discount_titles
(title_id   varchar(6)   default "PS7777" not null
        unique clustered
        references titles(title_id)
        check (title_id like "PS%"),
            new_price   money)
```

Для столбца title_id в таблице discount_titles определены все ограничения целостности.

- Вы можете создать сообщение об ошибке и привязать его к ограничениям ссылочной целостности и check . Для создания сообщений об ошибках используется процедура sp_addmessage, а для привязки к ограничениям – sp_bindmsg. Дополнительную информацию см. в описании системных процедур sp_addmessage и sp_bindmsg.
- Adaptive Server сначала проверяет данные на соответствие ограничениям check, а затем – ограничениям ссылочной целостности. Триггеры запускаются только после проверки всех ограничений целостности. Если какое-либо из ограничений не выполняется, Adaptive Server отменяет оператор изменения данных и не запускает триггеры. Тем не менее, при нарушении ограничений отката текущей транзакции *не* происходит.
- В родительской таблице нельзя обновить значения столбцов или удалить строки, которые соответствуют значениям в дочерней таблице. Сначала необходимо обновить или удалить строки в дочерней таблице, а затем – в родительской.
- Дочернюю таблицу необходимо удалять перед родительской, иначе ограничение будет нарушено.
- Получить информацию об ограничениях, определенных для таблицы, можно с помощью процедуры sp_helpconstraint.

Ограничение unique и ограничение первичного ключа

- Ограничения unique объявляются на уровне столбца и таблицы. Ограничения unique требуют, чтобы все значения в указанных столбцах были уникальными. Другими словами, любая пара строк таблицы не может содержать одинаковых значений в указанном столбце.

- Ограничение primary key является более сильным, чем ограничение unique. Столбцы с ограничением primary key не допускают значения NULL.

  Примечание. Ограничения unique и primary key оператора create table создают индексы, которые определяют столбцы с уникальными значениями и столбцы первичного ключа. Системные процедуры sp_primarykey, sp_foreignkey и sp_commonkey определяют логические отношения между столбцами. Эти отношения должны реализовываться с помощью индексов и триггеров.

- Ограничения unique и primary key уровня таблицы объявляются в операторе create table отдельными инструкциями и должны содержать имена одного или нескольких столбцов таблицы.

- Ограничения unique и primary key создают уникальный индекс по указанным столбцам. Ограничение unique в примере 3 создает уникальный кластерный индекс, который можно было бы создать следующим оператором:

  ```
  create unique clustered index salesind
      on sales (stor_id, ord_num)
  ```

  Единственным отличием является имя индекса. Однако при создании индекса с помощью ограничения можно также дать индексу имя salesind, если назвать ограничение соответствующим образом.

- В соответствии со стандартом SQL столбцы с ограничением unique не допускают значения NULL. По умолчанию (если в определении столбца не было указано ни null, ни not null) столбцы в Adaptive Server не допускают значения NULL (если установка по умолчанию не была изменена процедурой sp_dboption). В Transact-SQL столбцы с ограничением unique могут допускать значения NULL, так как уникальный индекс позволяет вставлять значения NULL.

- По умолчанию ограничения unique создают уникальные некластерные индексы, а ограничения primary key – уникальные кластерные индексы. Так как у таблицы может быть только один кластерный индекс, можно объявить только одно ограничение unique clustered или primary key clustered.

- Ограничения unique и primary key оператора create table являются упрощенной альтернативой оператору create index. Однако они имеют следующие ограничения:

  - с их помощью нельзя создавать неуникальные индексы;

- они не поддерживают некоторые параметры оператора create index;
- удалять эти индексы необходимо с помощью команды alter table drop constraint.

Ограничения ссылочной целостности

- Ограничения ссылочной целостности требуют, чтобы данные, вставляемые в *дочернюю* таблицу, совпадали со значениями в *родительской* таблице. Таблица удовлетворяет ограничению ссылочной целостности, если выполнено хотя бы одно из следующих условий:
  - данные в столбцах дочерней таблицы, ссылающихся на столбцы родительской таблицы, содержат значения NULL;
  - данные в столбцах дочерней таблицы, ссылающихся на столбцы родительской таблицы, совпадают с какими-либо значениями в соответствующих столбцах родительской таблицы.

  Например, строка, вставляемая в таблицу salesdetail (которая содержит сведения о продаже книг) базы данных pubs2, должна содержать допустимый идентификатор title_id из таблицы titles. Таблица salesdetail является дочерней, а таблица titles – родительской. В данный момент ссылочная целостность в базе данных pubs2 поддерживается при помощи триггеров. Однако для решения этой задачи можно определить в таблице salesdetail ограничение ссылочной целостности:

  ```
  title_id tid
      references titles(title_id)
  ```

- Один запрос может ссылаться не более чем на 192 таблицы. Для проверки ограничений ссылочной целостности, определенных на таблице, используется процедура sp_helpconstraint.
- Таблица может содержать ограничение ссылочной целостности по отношению к самой себе (то есть быть одновременно и родительской, и дочерней таблицей в ограничении). Например, в таблице store_employees базы данных pubs3, хранящей сведения о сотрудниках и их руководителях, может быть определено следующее ограничение ссылочной целостности между столбцами emp_id и mgr_id:

  ```
  emp_id id primary key,
  mgr_id id null
          references store_employees(emp_id),
  ```

  Это ограничение гарантирует, что все руководители являются служащими и каждому служащему будет назначен правильный руководитель.

- Чтобы удалить родительскую таблицу, нужно сначала удалить дочернюю таблицу или соответствующее ограничение ссылочной целостности (если только родительская таблица не является одновременно и дочерней).
- Adaptive Server не поддерживает ограничения ссылочной целостности для временных таблиц.
- Для создания таблицы, которая будет ссылаться на таблицу другого пользователя, нужно обладать полномочиями references для родительской таблицы. О том, как предоставлять полномочия references, см. в описании команды grant.
- Ограничения ссылочной целостности уровня таблицы объявляются в операторе create table отдельными инструкциями. Они должны содержать инструкцию foreign key и список из одного или нескольких имен столбцов.

  Имена столбцов в инструкции references указывать необязательно, если столбцы в родительской таблице объявлены первичным ключом инструкцией primary key.

  Столбцы родительской таблицы, на которые ссылается внешний ключ, должны быть ограничены уникальным индексом. Уникальный индекс можно создать либо при определении ограничения unique, либо явно с помощью оператора create index.
- Типы данных столбцов дочерней и родительской таблицы должны совпадать. Так, в приведенном ниже фрагменте кода тип данных столбца col1 в дочерней таблице (test_type) совпадает с типом данных столбца pub_id в родительской таблице (publishers):

  ```
  create table test_type
  (col1 char(4) not null
      references publishers(pub_id),
  col2 varchar(20) not null)
  ```
- Родительская таблица должна существовать на момент определения ограничения ссылочной целостности. Для создания таблиц, ссылающихся друг на друга, нужно определить эти таблицы одновременно с помощью оператора create schema. Кроме того, можно сначала создать обе таблицы без ограничений, а потом определить ограничения оператором alter table. Дополнительную информацию см. в описаниях команд create schema и alter table.
- Ограничения ссылочной целостности оператора create table предоставляют простой способ обеспечения целостности данных. В отличие от триггеров, они *не могут*:

- выполнять каскадные изменения в связанных между собой таблицах базы данных;
- реализовывать сложные ограничения (ссылаться на другие столбцы и объекты базы данных);
- выполнять анализ “что если”.

Ограничения ссылочной целостности не выполняют откат всей транзакции, когда какое-либо изменение данных их нарушает. Триггеры предоставляют свободу выбора в этом отношении: их можно написать так, чтобы они выполняли откат транзакций, нарушающих ограничения, а можно так, чтобы они продолжали такие транзакции.

Примечание. Adaptive Server проверяет ограничения ссылочной целостности перед выполнением всех триггеров, поэтому при изменении данных, нарушающем эти ограничения, триггеры не запускаются.

Использование межбазовых ограничений ссылочной целостности

- Если создается межбазовое ограничение (то есть ограничение, связывающее таблицы, расположенные в разных базах данных), в системных таблицах sysreferences обеих баз данных сохраняется следующая информация:

Таблица 7-17. Информация об ограничениях ссылочной целостности, хранящаяся в системных таблицах

Информация, хранящаяся в таблице sysreferences	Столбцы с информацией о родительской таблице	Столбцы с информацией о дочерней таблице
Идентификаторы ключевых столбцов	с refkey1 по refkey16	с fokey1 по fokey16
Идентификатор таблицы	reftabid	tableid
Идентификатор базы данных	pmrydbid	frgndbid
Имя базы данных	pmrydbname	frgndbname

- Дочернюю таблицу или базу данных, в которой она находится, можно удалить без проблем. Adaptive Server автоматически удаляет информацию о внешнем ключе из родительской базы данных.
- Поскольку дочерняя таблица зависит от информации из таблицы, на которую она ссылается, нельзя выполнять следующие действия:
  - удалять родительскую таблицу;

- удалять внешнюю базу данных, содержащую родительскую таблицу;
- переименовывать какую-либо из баз данных, участвующих в ограничении, с помощью процедуры sp_renamedb.

Перед выполнением любой из этих операций необходимо удалить межбазовое ограничение оператором alter table.

- Каждый раз при добавлении или удалении межбазового ограничения или при удалении таблицы, содержащей такое ограничение, необходимо сделать резервные копии *обеих* баз данных.

Предупреждение. Загрузка предыдущих версий базы данных, содержащей межбазовые ограничения, может привести к ее повреждению.

- *Имя* и идентификатор внешней базы данных хранятся в системной таблице sysreferences. Adaptive Server не гарантирует ссылочную целостность, когда база данных переименовывается или загружается на другой сервер командой load database.

Предупреждение. Перед созданием резервной копии базы данных, предназначенной для ее переименования или перемещения на другой сервер Adaptive Server, удалите все внешние ограничения ссылочной целостности командой alter table.

Ограничения *check*

- Ограничение check определяет, какие значения пользователь может ввести в столбец таблицы. В ограничении check указывается *условие_поиска*, на соответствие которому будут проверяться все значения, отличные от NULL, перед вставкой в таблицу. *Условие_поиска* может содержать:
  - список постоянных выражений, предваряемый ключевым словом in;
  - диапазон постоянных выражений, предваряемый ключевым словом between;
  - множество условий, заданных ключевым словом like, которые могут содержать метасимволы.

  Выражения могут содержать арифметические операции и встроенные функции Transact-SQL. *Условие_поиска* не может содержать подзапросы, агрегатные функции, переменные базового языка и параметры. В Adaptive Server не поддерживаются ограничения check для временных таблиц.

- Ограничение check, объявленное на уровне столбца, может ссылаться только на этот столбец, но ни на какие другие столбцы таблицы, тогда как ограничение check, объявленное на уровне таблицы, может ссылаться на любой столбец этой таблицы.
- Оператор create table позволяет объявить несколько ограничений check в определении столбца.
- Ограничение целостности check является альтернативой правилам и триггерам. Правила и триггеры относятся только к таблице, в которой они созданы, и их нельзя назначить столбцам других таблиц и пользовательским типам данных.
- Ограничение check не переопределяет определения столбцов. Если ограничение check объявлено для столбца, допускающего значения NULL, в этот столбец можно будет явно или неявно вставлять значения NULL, даже если значение NULL не входит в *условие_поиска*. Например, если ограничение check для столбца, допускающего значения NULL, имеет вид “pub_id in (”1389”, “0736”, “0877”, “1622”, “1756”)” или “@amount > 10000”, в такой столбец все равно можно будет вставить значения NULL. Определение столбцов имеет более высокий приоритет, чем ограничение check.

Столбцы IDENTITY

- При первой вставке значения в таблицу столбцу IDENTITY присваивается значение 1. При каждой следующей вставке значение увеличивается на 1. Это значение имеет приоритет над значением по умолчанию, объявленным для столбца в операторе create table или назначенным процедурой sp_bindefault. Максимальное значение, которое может быть вставлено в столбец IDENTITY, равно 10 $^{\text{точность}}$ - 1.
- С помощью вставки значения в столбец IDENTITY можно указать начальное значение для этого столбца или восстановить строку, удаленную по ошибке. Только владелец таблицы, владелец базы данных и системный администратор могут явно вставить значение в столбец IDENTITY после выполнения команды set identity_insert имя_таблицы on над базовой таблицей. Если по столбцу IDENTITY не построен уникальный индекс, СУБД Adaptive Server не проверяет уникальность значений. В столбец можно вставить любое положительное целое значение.
- Для ссылки на столбец IDENTITY можно использовать ключевое слово syb_identity с указанием имени таблицы вместо реального имени столбца.
- Чтобы в новые таблицы автоматически включался 10-значный столбец IDENTITY, системный администратор может установить пара-

метр базы данных auto identity в true с помощью следующей процедуры:

```
sp_dboption имя_базы_данных, "auto identity", "true"
```

Когда пользователь создает таблицу, не указывая ограничения primary key, unique или не создавая столбец IDENTITY, в таблице автоматически определяется столбец IDENTITY. Этот столбец (SYB_IDENTITY_COL) невидим при извлечении данных оператором select *. Чтобы вывести этот столбец, его необходимо явно включить в список выборки.

- В результате сбоев сервера между значениями столбца IDENTITY могут возникнуть разрывы. Эти разрывы могут также появиться при откате транзакций, удалении строк или вставке значений в столбец IDENTITY вручную. Максимальное значение этого разрыва (шага) зависит от значений параметров конфигурации identity burning set factor и identity grab size, а также от значений параметра identity_gap, указанного в операторах create table или select into. Дополнительную информацию о различных методах установки шага между значениями IDENTITY см. в разделе “Managing Identity Gaps in Tables” главы 7, “Creating Databases and Tables”, книги *Transact-SQL User’s Guide*.

Определение схемы блокировки

- Для определения схемы блокировки таблицы используется ключевое слово lock и один из следующих параметров:
  - allpages – блокировка страниц данных и индекса, к которым происходит обращение при выполнении запроса;
  - datapages – блокировка только страниц данных;
  - datarows – блокировка только строк данных.

  Если схема блокировки не указана, используется схема блокировки по умолчанию, заданная на уровне сервера (определяется значением параметра конфигурации lock scheme).
- Схема блокировки таблицы изменяется командой alter table.

Свойства управления пространством

- Свойства управления пространством fillfactor, max_rows_per_page, exp_row_size и reservepagegap используются следующим образом:
  - Параметр fillfactor оставляет дополнительное свободное пространство на страницах при создании индекса, однако когда индекс создан, размер свободного пространства больше не привязан к значению параметра fillfactor и может изменяться.

- Параметр max_rows_per_page ограничивает количество строк на странице данных или индекса. Он позволяет увеличить производительность параллельного доступа в таблицах с блокировкой всех страниц, так как уменьшение количества строк может уменьшить количество конфликтов блокировок. Если указать значение параметра max_rows_per_page и схему блокировки datapages или datarows, появится предупреждающее сообщение. Таблица будет создана, а значение параметра сохранится в таблице sysindexes, но оно будет использоваться только тогда, когда схема блокировки изменится на allpages.
- Параметр exp_row_size определяет ожидаемый размер строки данных. Он применяется только к строкам данных, но не к индексам, и только для таблиц с блокировкой только данных, содержащих столбцы переменной длины. Он позволяет сократить количество перемещенных строк в таблице с блокировкой только данных. В основном он нужен в таблицах, в которые при вставке заносятся пустые или короткие значения, а затем размер этих значений увеличивается при последующих обновлениях. Параметр exp_row_size резервирует пространство на странице данных на случай увеличения строки до указанного размера. Если указать параметр exp_row_size при создании таблицы с блокировкой всех страниц, появится предупреждающее сообщение. Таблица будет создана, а значение параметра сохранится в таблице sysindexes, но оно будет использоваться только тогда, когда схема блокировки изменится на datapages или datarows.
- Параметр reservepagegap определяет отношение пустых страниц к заполненным страницам для команд выделения экстентов. Он применяется к страницам данных и индексам при всех схемах блокировки.

- В таблице 7-18 перечислены допустимые комбинации свойств управления пространством и схемы блокировки. Если оператор create table содержит недопустимую комбинацию, появляется предупреждающее сообщение, но таблица будет создана. Значения сохраняются в системных таблицах, но не применяются. Значения вступают в силу, если в результате изменения схемы блокировки комбинация становится допустимой.

Таблица 7-18. Свойства управления пространством и схемы блокировки

Свойство	*allpages*	*datapages*	*datarows*
max_rows_per_page	Да	Нет	Нет
exp_row_size	Нет	Да	Да
reservepagegap	Да	Да	Да

Свойство	*allpages*	*datapages*	*datarows*
fillfactor	Да	Да	Да

- В таблице 7-19 перечислены значения по умолчанию и их влияние на свойства управления пространством.

Таблица 7-19. Значения по умолчанию и действие свойств управления пространством

Свойство	Значение по умолчанию	Результаты использования значения по умолчанию
max_rows_per_page	0	Размещает как можно больше строк на странице, максимальное количество равно 255.
exp_row_size	0	Использует значение по умолчанию уровня сервера, заданное параметром конфигурации default exp_row_size percent.
reservepagegap	0	Не оставляет пустые страницы при выделении экстентов.
fillfactor	0	Полностью заполняет листовые страницы, оставляя пространство на страницах индекса.

Использование параметра *exp_row_size*

- Если приложение вставляет короткие строки в таблицу с блокировкой только данных, а потом обновляет их, и при этом длина строк увеличивается, то необходимо использовать параметр exp_row_size, чтобы сократить количество перемещений строк в новое местоположение.

Использование параметра *reservepagegap*

- Команды, которым требуется большой объем пространства, выделяют пространство экстентами, а не отдельными страницами. Ключевое слово reservepagegap заставляет эти команды оставлять пустые страницы, чтобы страницы, выделенные впоследствии, находились рядом с расщепляемыми страницами и страницами, являющимися исходным местоположением перемещаемых строк. Использование ключевого слова reservepagegap описано в таблице 7-20.

Таблица 7-20. Применение параметра reservepagegap

Команда	Применяется к страницам данных	Применяется к страницам индекса
Быстрая bcp	Да	Быстрая bcp не применяется, если существуют индексы
Медленная bcp	Только для таблиц с неупорядоченными строками, не для таблиц с кластерным индексом	Выделение экстентов не выполняется

Команда	Применяется к страницам данных	Применяется к страницам индекса
select into	Да	Таблица, в которую копируются данные, не содержит индексов
create index или alter table...constraint	Да, для кластерных индексов	Да
reorg rebuild	Да	Да
alter table...lock (Из таблиц с блокировкой всех страниц в таблицы с блокировкой только данных или наоборот)	Да	Да

- Значение параметра reservepagegap для таблицы хранится в системной таблице sysindexes и применяется при выполнении любой из вышеперечисленных команд. Это значение можно изменить процедурой sp_chgattribute.
- Параметр reservepagegap не распространяется на рабочие таблицы и сортировки в рабочих таблицах.

Использование ключевого слова *at*

- В инструкции at задается та же информация о расположении, что и в системной процедуре sp_addobjectdef. Эта информация хранится в таблице sysattributes.

Столбцы Java-SQL

- Если в базе данных включена поддержка Java, можно создать таблицу со столбцами Java-SQL. Дополнительную информацию см. в книге *Java in Adaptive Server Enterprise*.
- Объявленный класс (*тип_данных*) столбца Java-SQL должен реализовывать интерфейс Serializable или Externalizable.
- При создании таблицы столбец Java-SQL нельзя:
  - указывать во внешнем ключе;
  - указывать в инструкции references;
  - определять с ограничением unique;
  - определять в качестве первичного ключа.
- Инструкция in row ограничивает значение столбца до 16 КБ, в зависимости от размера страницы сервера базы данных и других переменных.
- Если указана инструкция off row:
  - на этот столбец нельзя ссылаться в ограничении check;

- этот столбец нельзя указывать в операторе select с инструкцией distinct;
- этот столбец нельзя указывать в операторе сравнения, предикате и инструкции group by.

Получение информации о таблицах

- Процедура sp_help отображает информацию о таблицах, включая список атрибутов (например, привязка к кэшу) самой таблицы и ее индексов с указанием класса, имени, целочисленного и символьного значений и комментариев.
- Процедура sp_depends отображает информацию о представлениях, триггерах и процедурах базы данных, зависящих от таблицы.
- Процедура sp_helpindex отображает информацию об индексах, построенных по таблице.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Ниже перечислены расширения Transact-SQL:

- Использование имени базы данных при определении имен таблицы или столбцов
- Столбцы IDENTITY
- Использование столбцов not null по умолчанию
- Параметры asc и desc
- Параметр reservepagegap
- Инструкция lock
- Инструкция on имя_сегмента

  См. главу 1, “Системные и пользовательские типы данных”, или информацию о совместимости типов данных.

Полномочия

По умолчанию полномочия на выполнение команды create table принадлежат владельцу базы данных, который может передать их другим пользователям. Временные таблицы может создавать любой пользователь.

См. также

Команды alter table, create existing table, create index, create rule, create schema, create view, drop index, drop rule, drop table

Системные процедуры sp_addmessage, sp_addsegment, sp_addtype, sp_bindmsg, sp_chgattribute, sp_commonkey, sp_depends, sp_foreignkey, sp_help, sp_helpjoins, sp_helpsegment, sp_primarykey, sp_rename, sp_spaceused

create trigger

Описание

Создает триггер – особый тип хранимой процедуры, используемой для обеспечения ссылочной целостности. Триггер автоматически запускается при изменении данных в указанной таблице.

Синтаксис

```
create trigger [владелец .]имя_триггера
        on [владелец .]имя_таблицы
        for {insert , update , delete}
        as SQL_оператор
```

Или с инструкцией if update:

```
create trigger [владелец .]имя_триггера
    on [владелец .]имя_таблицы
    for {insert , update}
    as
        [if update (имя_столбца )
            [{and | or} update (имя_столбца )]...]
            SQL_операторы
        [if update (имя_столбца )
            [{and | or} update (имя_столбца )]...
            SQL_операторы
```

Параметры

имя_триггера

Имя триггера. Должно быть допустимым идентификатором и уникальным в базе данных. Чтобы создать триггер с тем же именем, что и существующий триггер, принадлежащий другому пользователю в текущей базе данных, перед именем триггера нужно указать имя владельца. По умолчанию *владельцем* является текущий пользователь. Если при определении триггера указано имя владельца, то перед именем таблицы также нужно указать имя таблицы.

Для указания имени триггера нельзя использовать переменную.

имя_таблицы

Имя таблицы, для которой создается триггер. Если в базе данных существует несколько таблиц с таким именем, необходимо указать имя владельца. По умолчанию *владельцем* является текущий пользователь.

insert, update, delete

При создании триггера эти параметры можно указывать в любой комбинации. Параметр delete нельзя указывать с инструкцией if update.

SQL_операторы

Определяют условия срабатывания и действия триггера. Условия срабатывания определяют, будут ли выполнены действия триггера в ответ на операции insert, update или delete. SQL-операторы часто содержат подзапросы, начинающиеся с ключевого слова if. В примере 2 подзап-

рос, идущий за ключевым словом if, является условием срабатывания триггера.

Действия триггера выполняются, когда пользователь пытается выполнить одну из операций insert, update или delete. Несколько действий триггера объединяются командами begin и end.

Список операторов, которые запрещено использовать в триггерах, см. в разделе "'Триггеры и транзакции'". Информацию о логических таблицах deleted и inserted, которые можно указывать в определении триггеров, см. в разделе "Логические таблицы deleted и inserted" на стр. 438.

if update

Проверяет, входит ли указанный столбец в список инструкции set оператора update или было ли в него вставлено значение оператором insert. Эта инструкция позволяет выполнять действия только при изменении указанного столбца (см. пример 3). При создании триггера оператором create trigger можно указать несколько столбцов, объявив несколько инструкций if update (см. пример 5).

Примеры

Пример 1. Создание триггера, выводящего сообщение всякий раз, когда кто-либо пытается добавить или изменить данные в таблице titles:

```
create trigger reminder
on titles
for insert, update as
print "Don't forget to print a report for accounting."
```

Пример 2. Создание триггера, запрещающего вставку строк в таблицу titleauthor, если в таблице titles нет соответствующего идентификатора title_id:

```
create trigger t1
on titleauthor
for insert as
if (select count(*)
    from titles, inserted
    where titles.title_id = inserted.title_id) = 0
begin
print "Please put the book's title_id in the
        titles table first."
rollback transaction
end
```

Пример 3. Создание триггера, вносящего соответствующие изменения в таблицу titles при изменении столбца pub_id в таблице publishers:

```
create trigger t2
on publishers
```

```
for update as
if update (pub_id) and @@rowcount = 1
begin
    update titles
    set titles.pub_id = inserted.pub_id
    from titles, deleted, inserted
    where deleted.pub_id = titles.pub_id
end
```

Пример 4. Создание триггера, удаляющего строки с информацией о книге из таблицы titles при удалении любой строки из таблицы titleauthor. Если книга написана несколькими авторами, то другие ссылки на нее в таблице titleauthor также удаляются:

```
create trigger t3
on titleauthor
for delete as
begin
    delete titles
    from titles, deleted
    where deleted.title_id = titles.title_id
    delete titleauthor
    from titleauthor, deleted
    where deleted.title_id = titleauthor.title_id
    print "All references to this title have been
    deleted from titles and titleauthor."
end
```

Пример 5. Создание триггера, запрещающего обновление первичного ключа по выходным. Он также запрещает обновления столбцов price (цена) и advance (аванс), в результате которых общая выручка (произведение столбцов total_sales и price) становится меньше размера аванса:

```
create trigger stopupdatetrig
on titles
for update
as
if update (title_id)
  and datename(dw, getdate())
  in ("Saturday", "Sunday")
  begin
    rollback transaction
    print "We don't allow changes to"
    print "primary keys on the weekend!"
  end
if update (price) or update (advance)
  if (select count(*) from inserted
    where (inserted.price * inserted.total_sales)
    < inserted.advance) > 0
    begin
    rollback transaction
```

```
        print "We don't allow changes to price or"
        print "advance for a title until its total"
        print "revenue exceeds its latest advance."
        end
```

Использование

- Триггер срабатывает только один раз для каждого изменения данных. Сложные запросы, содержащие цикл while, могут выполнять операторы update или insert несколько раз, и каждый раз запускается триггер.

Триггеры и ссылочная целостность

- Триггеры обычно используются для обеспечения *ссылочной целостности* (правила целостности об отношениях между первичными и внешними ключами таблиц и представлений), выполнения каскадных удалений и обновлений (см. примеры 2, 3 и 4).
- Триггер запускается только после того, как оператор, изменяющий данные, завершится и сервер Adaptive Server проверит, есть ли нарушения типов данных, правил и ограничений целостности. Триггер и запустивший его оператор являются единой транзакцией, и триггер может выполнить откат этой транзакции. Если при выпол-нении триггера возникает ошибка, происходит откат всей транзакции.
- Для обеспечения ссылочной целостности можно определить ограничения с помощью команды create table, не прибегая к созданию триггеров. Об ограничениях целостности см. в описании команд create table и alter table.

Логические таблицы *deleted* и *inserted*

- Таблицы deleted и inserted являются логическими (абстрактными). Их структура совпадает со структурой таблицы, для которой определен триггер. Эти таблицы содержат старые и новые строки, изменившиеся в результате действия пользователя.
- Содержимое таблиц deleted и inserted может просматриваться триггерами и влиять на их выполнение, но не может быть изменено действиями триггера.
- Таблица deleted используется операторами delete и update, а таблица inserted – операторами insert и update. Операция update фактически состоит из двух операций: delete и insert, сначала она изменяет таблицу deleted, а затем – таблицу inserted.

Ограничения на триггеры

- Триггер можно создать только в текущей базе данных. Если при обращении к триггеру указано имя владельца, то оно также должно быть указано в имени таблицы. Триггер может ссылаться на объекты в другой базе данных.

- Триггер не может применяться к более чем одной таблице. Однако в одном операторе create trigger можно определить действие триггера для нескольких событий (например, insert и update). Для каждой таблицы можно создать до трех триггеров – по одному для каждой из операций insert, update и delete.
- Если триггер создается для операции (insert, update или delete), для которой уже был определен триггер, новый триггер заменит старый. Перед заменой триггера не появляется предупреждающее сообщение.
- Триггер нельзя создать для временной таблицы.
- Триггер нельзя создать для представления.
- Триггер нельзя создать для системной таблицы.
- Триггеры не могут извлекать данные из столбцов типа text и image таблиц inserted и deleted.
- Sybase не рекомендует включать в триггеры операторы select, возвращающие результат пользователю, так как это должно выполняться приложениями, изменяющими данные в таблице.
- Если триггер ссылается на таблицы, столбцы или представления, имена которых не являются допустимыми идентификаторами, необходимо выполнить команду set quoted_identifier on перед выполнением команды create trigger и заключать такие имена в двойные кавычки. Однако параметр quoted_identifier *не обязательно* должен быть установлен в on при срабатывании триггера.

Триггеры и производительность

- Триггеры не оказывают большого влияния на производительность. Время выполнения триггера складывается в основном из времени на обращение к другим таблицам, которые находятся в памяти или на устройстве базы данных.
- Поскольку таблицы deleted и inserted, к которым часто обращаются триггеры, являются логическими, они всегда находятся в памяти, а не на устройстве базы данных. Затраты времени на выполнение операций определяются расположением других таблиц, к которым обращаются триггеры.

Установка параметров внутри триггеров

- В триггере можно выполнять команду set. Параметр, установленный этой командой set, действует во время выполнения триггера, а затем возвращается к исходному значению. В частности, внутри триггера можно установить параметр self_recursion, чтобы изменения данных, сделанные триггером, снова запускали этот триггер.

Удаление триггера

- При удалении или переименовании объектов, на которые ссылается триггер, необходимо удалить его и создать повторно. Для переименования триггеров используется процедура sp_rename.
- Когда таблица удаляется, все связанные с ней триггеры также удаляются.

Действия, не вызывающие запуск триггеров

- Команда truncate table не вызывает срабатывания триггера delete. Несмотря на то что команда truncate table выполняет то же действие, что команда delete без инструкции where (удаляет все строки), сделанные ею изменения строк данных не записываются в журнал, поэтому триггер не запускается.

  Так как полномочия на выполнение команды truncate table принадлежат владельцу таблицы и не могут быть переданы другим пользователям, только владелец таблицы может удалить строки этой командой, не вызвав триггер.
- Команда writetext не вызывает срабатывания триггера, независимо от того, записывается ли она в журнал или нет.

Триггеры и транзакции

- Действия, которые выполняются при срабатывании триггера, всегда неявно являются частью транзакции вместе с самим триггером. Триггеры часто реализованы так, чтобы выполнять откат всей транзакции при обнаружении ошибок. Тем не менее, в следующих случаях они выполняют откат только определенных изменений данных:
  - Если триггер содержит команду rollback transaction, то производится откат всего пакета и последующие операторы этого пакета не выполняются.
  - Если триггер содержит команду rollback trigger, выполняется откат только тех изменений, которые вызвали срабатывание триггера. Команда rollback trigger может содержать оператор raiserror. В этом случае последующие операторы пакета будут выполнены.
- Так как триггеры являются частью транзакции, в них нельзя использовать следующие операторы и системные процедуры:
  - все команды create, включая create database, create default, create index, create procedure, create rule, create table, create trigger и create view;

- команды drop;
- alter database и alter table
- truncate table
- grant и revoke
- update statistics
- sp_configure;
- load database и load transaction
- disk init, disk mirror, disk refit, disk reinit, disk remirror, disk unmirror
- select into

- Если желаемый результат (например, итоговое значение) зависит от количества строк, измененных оператором, то можно использовать переменную *@@rowcount*, чтобы определить количество вставленных, измененных или удаленных строк (для команд insert, delete или update, основанных на операторе select), и выполнить соответствующие действия. Любой оператор Transact-SQL, который не возвращает строки (например, if), присваивает переменной *@@rowcount* значение 0, поэтому переменная *@@rowcount* должна проверяться в начале триггера.

Триггеры вставки и обновления

- Когда выполняется команда insert или update, Adaptive Server добавляет строку в таблицу триггера и в таблицу inserted. Строки в таблице inserted всегда совпадают с одной или несколькими строками в таблице триггера.
- Триггеры update или insert могут использовать команду if update, чтобы определить, был ли определенный столбец изменен командой update или insert. Условие if update(имя_столбца) возвратит true, если триггер был вызван оператором insert, в котором для данного столбца было задано значение (в списке выборки или инструкции values). Если для столбца было явно указано значение NULL или значение по умолчанию, триггер также будет активирован. Однако неявное присваивание значений NULL не активирует триггер.

  Например, пусть есть таблица и триггер, созданные следующими операторами:

```
create table junk
(aaa int null,
bbb int not null)
```

```
create trigger trigtest on junk
for insert as
if update (aaa)
    print "aaa updated"
if update (bbb)
    print "bbb updated"
```

Вставка значений в любой из столбцов или в оба столбца вызывает триггер для обоих столбцов aaa и bbb:

```
insert junk (aaa, bbb)
values (1, 2)
aaa updated
bbb updated
```

Явная вставка значения NULL в столбец aaa также приводит к срабатыванию триггера:

```
insert junk
values (NULL, 2)
aaa updated
bbb updated
```

Если для столбца aaa определено значение по умолчанию, триггер также сработает.

Однако если у столбца aaa нет значения по умолчанию и при вставке для него не было указано значение, ему неявно будет присвоено значение NULL и триггер не сработает:

```
insert junk (bbb)
values(2)
bbb updated
```

Условие if update никогда не бывает истинно для оператора delete.

Вложенные триггеры и рекурсия

- По умолчанию Adaptive Server поддерживает вложенные триггеры. Для запрещения вложенности триггеров необходимо установить параметр allow nested triggers в 0 (отключено) процедурой sp_configure, как показано ниже:

```
sp_configure "allow nested triggers", 0
```

- Вложенность триггеров не может превышать 16 уровней. Если триггер изменяет таблицу с другим триггером, срабатывает второй триггер, который может запустить третий триггер и т.д. Если в этой цепочке образуется замкнутый цикл, максимальный уровень вложен-

ности оказывается превышенным и триггер аварийно завершается, вызывая откат содержащей его транзакции.

Примечание. Так как триггеры являются частью транзакции, ошибка в любом триггере из цепочки вложенных триггеров вызывает откат всей транзакции. В тело триггера нужно включить сообщения и другие средства отладки и обработки ошибок, чтобы определить место, где возникла ошибка.

- Глобальная переменная *@@nestlevel* содержит текущий уровень вложенности. Каждый раз, когда хранимая процедура или триггер вызывают другую хранимую процедуру или триггер, уровень вложенности увеличивается на единицу. При превышении максимального уровня, равного 16, транзакция аварийно завершается.
- Если триггер вызывает хранимую процедуру, которая в свою очередь вызывает срабатывание триггера, триггер повторно активируется, только если разрешена вложенность триггеров. Если в триггере не предусмотрены условия, ограничивающие количество рекурсий, такой вызов триггеров приведет к превышению максимального уровня вложенности.

  Например, когда триггер обновления вызывает хранимую процедуру, которая выполняет обновление, они выполняются только один раз, если параметр allow nested triggers установлен в 0. Если параметр allow nested triggers установлен в 1 (включен) и количество обновлений не ограничено условием триггера или процедуры, эта цепочка вызовов будет продолжаться, пока не будет превышен максимальный уровень вложенности, равный 16.
- По умолчанию триггер, выполняющий изменение той же таблицы, для которой он был определен, не вызывает сам себя, независимо от значения параметра конфигурации allow nested triggers. Чтобы разрешить вызов триггерами самих себя, включите параметр self_recursion командой set. Например, если триггер, реагирующий на обновление одного столбца таблицы, обновляет другой столбец, триггер срабатывает только один раз, если параметр self_recursion отключен, но может сработать до 16 раз, если параметр self_recursion установлен в on. Параметр конфигурации allow nested triggers также должен быть установлен в on для разрешения саморекурсии.

Получение информации о триггерах

- План выполнения триггера хранится в таблице sysprocedures.

- Каждому триггеру назначается идентификатор, который отдельной строкой хранится в таблице sysobjects вместе с идентификатором таблицы, для которой он создан, в столбце deltrig, а также в виде записей в столбцах deltrig, instrig и updtrig таблицы sysobjects для таблицы, для которой он создан.
- Для отображения текста триггера, который хранится в таблице syscomments, используется процедура sp_helptext.

  Если администратор по безопасности выключил параметр allow select on syscomments.text column системной процедурой sp_configure (как этого требует оценочная конфигурация Adaptive Server), только создатель триггера и системный администратор смогут просмотреть его исходный текст процедурой sp_helptext.
- Для отображения информации о триггере используется процедура sp_help.
- Для отображения информации о таблицах и представлениях, на которые ссылается триггер, используется процедура sp_depends.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Только администратор безопасности может предоставлять и отзывать полномочия на создание триггеров. Владелец базы данных может создавать триггеры в любых пользовательских таблицах. Пользователи могут создавать триггеры только в собственных таблицах.

Полномочия на выполнение команды create trigger по умолчанию предоставлены всем пользователям.

Когда администратор безопасности отзывает у пользователя полномочия на создание триггера, в таблицу sysprotects добавляется строка, соответствующая этому отзыву. Для предоставления полномочий на выполнение оператора create trigger нужно выполнить две команды grant: первая команда удаляет из таблицы sysprotects строку, отвечающую за отзыв полномочий, а вторая вставляет строку, разрешающую выполнение оператора. Если полномочия на создание триггеров отозваны, пользователь не может создавать триггеры даже для тех таблиц, которыми он владеет. Полномочия на создание триггеров отзываются только в той базе данных, в которой была выполнена команда revoke.

Полномочия на объекты при создании триггера При создании триггера Adaptive Server не проверяет полномочия на объекты (например, на таблицы и представления), на которые ссылается этот триггер. Таким образом, триггер будет успешно создан, даже если у создающего его пользователя нет прав доступа к этим объектам. Все полномочия проверяются при запуске триггера.

Полномочия на объекты при выполнении триггера Механизм проверки полномочий на доступ к объектам, на которые ссылается этот триггер (выполняющийся при срабатывании триггера), зависит от того, принадлежат ли триггер и все эти объекты одному и тому же пользователю.

- Если тригер и объекты принадлежат разным пользователям, пользователь, который выполнил операцию, вызвавшую срабатывание триггера, должен обладать прямым доступом к объектам. Например, если триггер пытается сделать запрос к таблице, к которой у данного пользователя нет прав доступа, триггер не будет выполнен. Кроме того, для изменений данных, вызвавших срабатывание триггера, производится откат.
- Если триггер и объекты принадлежат одному и тому же пользователю, применяются специальные правила. Пользователь автоматически получает “неявные” полномочия на доступ к объектам триггера, даже если он не может обращаться к ним напрямую. Подробное описание правил для неявных полномочий см. в книге *Руководство по системному администрированию*.

См. также

Команды alter table, create procedure, create procedure, drop trigger, rollback trigger, set

Системные процедуры sp_commonkey, sp_configure, sp_depends, sp_foreignkey, sp_help, sp_helptext, sp_primarykey, sp_rename, sp_spaceused

create view

Описание

Создает представление данных из одной или нескольких таблиц.

Синтаксис

create view [*владелец* .]*имя_представления*
[(*имя_столбца* [, *имя_столбца*]...)]
as
select [distinct] *оператор_select*
[with check option]

Параметры

имя_представления

Имя представления. Не может содержать имя базы данных. Если параметр set quoted_identifier равен on, то имя представления может содержать разделители. В противном случае имя представления не может быть переменной и должно соответствовать правилам для идентификаторов. Дополнительную информацию о допустимых именах представлений см. в разделе “Идентификаторы” на стр. 242. Чтобы создать представление с тем же именем, что и существующее представление, принадлежащее другому пользователю в текущей базе данных, перед именем представления нужно указать имя владельца. По умолчанию *владелец* – текущий пользователь.

имя_столбца

Имена, которые будут использоваться в качестве заголовков столбцов в представлении. Если параметр set quoted_identifier равен on, то имена могут содержать разделители. В противном случае оно должно соответствовать правилам для идентификаторов. Информацию о допустимых именах столбцов см. в разделе “Идентификаторы” на стр. 242.

Имена столбцов можно указывать всегда, но требуются они только в следующих случаях:

- если
- столбец содержит результат арифметического выражения, функции, конкатенации строк или константу;
- если несколько столбцов имеют одинаковые имена (это часто бывает при соединениях);
- если нужно назначить столбцу в представлении другое имя (см. пример 3).

Имена столбцов можно также назначить в операторе select (см. пример 4). Если имена столбцов не указаны, столбцы представления получают имена, указанные в операторе select.

select

Начинает оператор select, определяющий представление.

distinct
Указывает, что представление не может содержать повторяющиеся строки.

оператор_select
завершает оператор select, определяющий представление. Оператор может ссылаться на несколько таблиц и другие представления.

with check option
Указывает, что операторы не могут изменять данные через представление так, чтобы новые данные не соответствовали условию запроса, определяющего представление. Другими словами, все строки, вставленные и обновленные через представление, должны по-прежнему попадать в это представление.

Примеры

Пример 1. Создание представления по столбцам title, type, price и pubdate базовой таблицы titles:

```
create view titles_view
as select title, type, price, pubdate
from titles
```

Пример 2. Создание представления newold view на основе представления old view. Оба столбца в новом представлении переименовываются. Все имена представлений и столбцов, содержащие пробелы, заключаются в двойные кавычки. Перед созданием этого представления необходимо выполнить команду set quoted_identifier on:

```
create view "new view" ("column 1", "column 2")
as select col1, col2 from "old view"
```

Пример 3. Создание представления, содержащего названия, авансы и суммы к получению для книг, цена которых выше $5.00:

```
create view accounts (title, advance, amt_due)
as select title, advance, price * total_sales
from titles
where price > $5
```

Пример 4. Создание представления на основе таблиц authors и publishers. Представление содержит имена авторов и названия городов, где они проживают, для тех авторов, которые живут в городе, где есть хотя бы один издатель:

```
create view cities
(authorname, acity, publishername, pcity)
as select au_lname, authors.city, pub_name,
publishers.city
from authors, publishers
where authors.city = publishers.city
```

Пример 5. Создание представления с тем же определением, что и в примере 3, с той разницей, что заголовки столбцов назначаются командой select:

```
create view cities2
as select authorname = au_lname,
acity = authors.city, publishername = pub_name, pcity =
publishers.city
from authors, publishers
where authors.city = publishers.city
```

Пример 6. Создание представления author_codes по таблице titleauthor, в котором перечислены уникальные идентификаторы авторов:

```
create view author_codes
as select distinct au_id
from titleauthor
```

Пример 7. Создание представления price_list по таблице title, в котором перечислены неповторяющиеся цены на книги:

```
create view price_list (price)
as select distinct price
from titles
```

Пример 8. Создание представления по таблице stores, в котором содержатся сведения о магазинах, находящихся только в Калифорнии (код "CA"). Инструкция with check option проверяет, удовлетворяют ли вставляемые и обновляемые строки условию выбора, определяющему представление. Строки, значение которых в столбце state не равно "CA", не принимаются:

```
create view stores_cal
as select * from stores
where state = "CA"
with check option
```

Пример 9. Создание представления stores_cal30 по представлению stores_cal. Новое представление наследует параметр with check option из представления stores_cal. Во всех строках, вставляемых или обновляемых через представление stores_cal30, значение столбца state должно равняться "CA". Однако поскольку в определении представления stores_cal30 не была указана инструкция with check option, то при вставке и обновлении строк через представление stores_cal30 столбцу payterms можно присваивать значение, отличное от "Net 30":

```
create view stores_cal30
as select * from stores_cal
where payterms = "Net 30"
```

Пример 10. Создание представления stores_cal30_check на основе представления stores_cal. Новое представление наследует параметр with check option из представления stores_cal. Кроме того, в его определении также указана инструкция with check option. Поэтому все строки, вставляемые или обновляемые через представление stores_cal30_check, должны удовлетворять условиям подзапросов в определениях представлений stores_cal и stores_cal30_check. Строки, в которых значение столбца state не равно "CA" или значение столбца payterms не равено "Net 30", не принимаются:

```
create view stores_cal30_check
as select * from stores_cal
where payterms = "Net 30"
with check option
```

Использование

- Представления можно использовать в качестве механизма защиты, предоставляя полномочия на доступ к представлениям, а не к таблицам.
- Для переименования представления используется процедура sp_rename.
- Когда над представлением выполняется оператор, Adaptive Server проверяет, существуют ли все объекты базы данных, на которые оно ссылается, допустимы ли они в контексте оператора и не нарушают ли команды обновления правил целостности данных. Если обнаружено хотя бы одно нарушение, выдается сообщение об ошибке. Если проверка заканчивается успешно, команда create view "преобразует" запрос к представлению в запрос к базовым таблицам.
- Дополнительную информацию о представлениях см. в книге *Transact-SQL User's Guide*.

Ограничения на представления

- Представление можно создать только в текущей базе данных.
- Количество столбцов, на которые ссылается представление, не может превышать 1024.
- Нельзя создать представление на основе временной таблицы.
- Для представления нельзя создать триггеры и индексы.
- Команды readtext и writetext нельзя выполнять над столбцами представления, имеющими тип text или image.
- Команда select, определяющая представление, не может содержать инструкций order by и compute и ключевого слова into.

- Через представление нельзя вставлять и обновлять строки, если определяющая его команда select содержит оператор union.
- Через представление нельзя удалять строки, если определяющая его команда select содержит оператор union.
- Команду create view можно объединять с другими SQL-командами в один пакет.

Предупреждение. Если команда create view находится внутри блока if...else или цикла while, Adaptive Server создает схему для представления перед проверкой истинности условия. Это может привести к ошибке, если представление уже существует. Поэтому нужно убедиться в том, что представление с таким именем отсутствует в базе данных.

- В команде create view нельзя использовать переменные:

```
declare @p int
select @p = 2
create view v2
as
select * from t1 where c1 > @p
```

При выполнении такой команды будет выдано сообщение об ошибке 7351 – “Local or global variables not allowed in view definition”.

Решение проблем, связанных с представлениями

- Если структура базовой таблицы представления, определенного командой select *, меняется (удаляются или добавляются столбцы), то для того, чтобы в этом представлении были видны новые столбцы, нужно его удалить и создать повторно. Интерпретация символа звездочки и формирование списка столбцов происходит при создании представления.
- Если представление зависит от таблицы или представления, которые были удалены, то при обращении к этому представлению будет выдано сообщение об ошибке. Если создать схему и новую таблицу (представление) с тем же именем, представление снова станет доступно.
- Представление можно переопределить, не изменяя зависимые представления, если после этого переопределения преобразование запросов, определяющих зависимые представления, по-прежнему будет возможным.

Изменение данных через представления

- Оператор delete запрещено использовать с многотабличным представлением.
- Оператор insert запрещено использовать в представлении, в которое входят не все столбцы базовой таблицы или представления, не допускающие значения NULL (Adaptive Server не может указать значения для столбцов базовой таблицы или представления, не допускающих значения NULL).
- Строку нельзя вставить через представление, содержащее вычисляемый столбец.
- Оператор insert запрещено выполнять над представлением с соединением, созданным с инструкцией distinct или with check option.
- Оператор update можно выполнять над представлением с соединением, созданным с инструкцией with check option. Обновление не выполняется, если любой из изменяемых столбцов присутствует в инструкции where в выражении, содержащем столбцы из нескольких таблиц.
- Чтобы строку можно было вставить или обновить через представление с соединением, все затрагиваемые этой операцией столбцы должны принадлежать одной и той же базовой таблице.
- Строку нельзя вставить или обновить в представлении, созданном с инструкцией distinct.
- Операторы обновления данных не изменяют вычисляемые столбцы представления и представления с агрегатными значениями.

Столбцы IDENTITY и представления

- Синтаксис *имя_столбца* = identity(*точность*) не позволяет добавить в представление новый столбец IDENTITY.
- Чтобы вставить явное значение в столбец IDENTITY, владелец таблицы, владелец базы данных или системный администратор должны выполнить команду set identity_insert имя_таблицы on для базовой таблицы столбца напрямую, а не через представление.

Инструкция *group by* и представления

- Если представление создается в целях защиты, нужно соблюдать осторожность при использовании агрегатных функций и инструкции group by. Расширение Transact-SQL позволяет объявлять столбцы, не указанные в инструкции group by. Если указан столбец, отсутству-

ющий в инструкции group by, будут возвращены все строки. Например, следующий запрос вернет все 18 строк – возможно, больше, чем требовалось:

```
select title_id, type, sum(total_sales)
from titles
group by type
```

В то время как следующий запрос вернет одну строку для каждого значения столбца type (6 строк):

```
select type, sum(total_sales)
from titles
group by type
```

Дополнительную информацию об инструкции group by см. в разделе “Инструкции group by и having на стр. 576”.

Инструкция *distinct* и представления

- Инструкция distinct определяет представление как объект базы данных, не содержащий повторяющихся строк. Строка считается повторением другой строки, если значения всех столбцов первой строки совпадают со значениями столбцов второй. Значения NULL считаются повторяющимися.

  Если запрос выбирает подмножество столбцов представления, некоторые строки могут казаться идентичными друг другу. Если некоторые комбинации столбцов, выбираемых запросом, содержат одни и те же значения, то будет казаться, что результирующий набор содержит повторяющиеся строки. Однако строки самого представления останутся уникальными. Adaptive Server применяет инструкцию distinct к определению представления при первом обращении к представлению (перед операциями проекции и выборки), поэтому все строки представления являются уникальными.

  При определении представления можно несколько раз указать инструкцию distinct в составе агрегатных функций или инструкции group by оператора select, чтобы избавиться от повторяющихся строк. Например:

  ```
  select distinct count(distinct title_id), price
  from titles
  ```

- Инструкция distinct распространяется только на это представление и не влияет на другие представления, образованные от него.

Параметр *with check option* и представления

- Если представление создано с параметром with check option, все строки, вставляемые или обновляемые через это представление, должны удовлетворять его условию отбора.
- Если представление создано с параметром with check option, все представления, которые от него образованы, должны удовлетворять его условию отбора. Все строки, вставленные и обновленные через производное представление, должны оставаться видимыми в этом представлении.

Получение информации о представлениях

- Для отображения информации о таблицах или представлениях, от которых зависит данное представление, и об объектах, которые на него ссылаются, используется процедура sp_depends.
- Для отображения текста представления, который хранится в таблице syscomments, выполните процедуру sp_helptext, указав имя представления в качестве параметра.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Использование нескольких ключевых слов distinct и выражения "*заголовок_столбца = имя_столбца*" в операторе select является расширением Transact-SQL.

Полномочия

По умолчанию полномочия на выполнение команды create view принадлежат владельцу базы данных, который может передать их другим пользователям.

Полномочия на объекты при создании представления При создании представления Adaptive Server не проверяет полномочия на объекты (таблицы и другие представления), на которые оно ссылается. Таким образом, представление будет успешно создано, даже если у создающего его пользователя нет прав доступа к базовым объектам представления. Все полномочия проверяются при вызове представления.

Полномочия на объекты при вызове представления Механизм проверки полномочий на использование объектов представления, выполняемой при вызове представления, зависит от того, принадлежат ли представление и все объекты, по которым оно построено, одному и тому же пользователю.

- Если представление и его базовые объекты принадлежат разным пользователям, то пользователю, вызвавшему представление, должен быть предоставлен прямой доступ к этим объектам. Например, если представление извлекает данные из таблицы, доступ к которой запрещен данному пользователю, выборка завершится ошибкой.

- Если представление и объекты принадлежат одному пользователю, применяются специальные правила. Пользователю, который вызывает представление, автоматически назначаются “неявные полномочия” доступа к объектам представления, даже если он не может обращаться к ним напрямую. С помощью представлений можно предоставить ограниченный доступ пользователям к данным таблиц, не предоставляя им прямого доступа к таблицам. Это позволяет использовать представление как механизм защиты. Например, с помощью представления можно предоставить пользователю доступ только к определенным столбцам и строкам таблицы. Подробное описание правил для неявных полномочий см. в книге *Руководство по системному администрированию*.

См. также

Команды create schema, drop view, update

Системные процедуры sp_depends, sp_help, sp_helptext, sp_rename

dbcc

Описание

Команда dbcc (Database Consistency Checker) проверяет логическую и физическую согласованность базы данных, а также предоставляет средства сбора статистических данных, планирования и устранения неполадок.

Синтаксис

dbcc checkalloc [(*имя_базы_данных* [, fix | nofix])]

dbcc checkcatalog [(*имя_базы_данных*)]

dbcc checkdb [(*имя_базы_данных* [, skip_ncindex])]

dbcc checkstorage [(*имя_базы_данных*)]

dbcc checktable({*имя_таблицы* | *идентификатор_таблицы*}[, skip_ncindex])

dbcc checkverify [(*имя_базы_данных*)]

dbcc complete_xact (*идентификатор_транзакции*, {"commit" | "rollback"})

dbcc forget_xact (*идентификатор_транзакции*)

dbcc dbrepair (*имя_базы_данных*, dropdb)

dbcc engine({offline , [*номер_экземпляра_сервера*] | "online" })

dbcc fix_text ({*имя_таблицы* | *идентификатор_таблицы*})

dbcc indexalloc ({*имя_таблицы* | *идентификатор_таблицы*}, *идентификатор_индекса*
[, {full | optimized | fast | null}
[, fix | nofix]])

dbcc rebuild_text (*таблица* [, *столбец*
[, *номер_текстовой_страницы*]])

dbcc reindex ({*имя_таблицы* | *идентификатор_таблицы*})

dbcc tablealloc ({*имя_таблицы* | *идентификатор_таблицы*}
[, {full | optimized | fast | null}
[, fix | nofix]])|

dbcc { traceon | traceoff } (*флаг* [, флаг ...])

dbcc tune ({ ascinserts, {0 | 1 } , *имя_таблицы* |
cleanup, {0 | 1 } |
cpuaffinity, *начальный_процессор* {, on| off } |
des_greedyalloc, *id_базы_данных*, *имя_объекта*,
" { on | off }" |
deviochar vdevno, "*размер_пакета*" |
doneinproc { 0 | 1 } |
maxwritedes, *число_операций_записи_на_пакет* })

Параметры

checkalloc

Проверяет, что все страницы в указанной базе данных выделены правильно и ни одна из выделенных страниц не осталась неиспользованной. Если имя базы данных не указано, параметр checkalloc проверяет текущую базу данных. Он всегда использует параметр отчета optimized (см. описание параметра tablealloc).

Параметр checkalloc выдает отчет об объеме выделенного и используемого пространства.

имя_базы_данных
Имя базы данных, подлежащей проверке. Если имя базы данных не указано, утилита dbcc проверяет текущую базу данных.

fix | nofix
Определяет, должна ли команда dbcc устранять обнаруженные ошибки выделения памяти. Режимом по умолчанию параметра checkalloc является nofix (не устранять). Для использования параметра fix необходимо перевести базу данных в однопользовательский режим.

Вопросы, связанные с выделением страниц на сервере Adaptive Server, рассматриваются в книге *Руководство по системному администрированию*.

checkcatalog
Проверяет согласованность как внутри системных таблиц, так и между системными таблицами. Например, если указан параметр checkcatalog, то будет проверено, что для каждого типа данных в таблице syscolumns имеется соответствующая строка в таблице systypes, что у каждой таблицы и представления, перечисленных в таблице sysobjects, имеется по меньшей мере один столбец в таблице syscolumns, и что последняя контрольная точка в таблице syslogs является действительной. При этом параметре также будет выдан отчет обо всех определенных сегментах. Если имя базы данных не указано, будет проверена текущая база данных.

checkdb
Выполняет в указанной базе данных те же проверки, что и параметр checktable, но для каждой таблицы (включая таблицу syslogs). Если имя базы данных не указано, параметр checkdb проверяет текущую базу данных.

skip_ncindex
Отменяет проверку некластерных индексов пользовательских таблиц в режимах dbcc checktable или dbcc checkdb. По умолчанию проверяются все индексы.

checkstorage
Проверяет выделение пространства в указанной базе данных, записи страниц OAM, согласованность страниц, текстовые столбцы, память, выделенную для текстовых столбцов, и цепочки текстовых столбцов. Результаты каждой операции dbcc checkstorage сохраняются в базе данных dbccdb. Подробную информацию об использовании команды dbcc checkstorage, а также о создании и сопровождении базы данных dbccdb и формировании отчетов по результатам проверок см. в книге *Руководство по системному администрированию*.

checktable

Проверяет в указанной таблице правильность связей индексных страниц и страниц данных, упорядоченность индексов, согласованность всех указателей, корректность данных на каждой странице и корректность смещений страниц. Если сегмент журнала расположен на специально выделенном для него устройстве, то при выполнении команды dbcc checktable над таблицей syslogs будет выдан отчет об используемом и свободном пространстве журнала (журналов), например:

```
Checking syslogs
The total number of data pages in this table is 1.
*** NOTICE: Space used on the log segment is 0.20 Mbytes, 0.13%.
*** NOTICE: Space free on the log segment is 153.4 Mbytes, 99.87%.

DBCC execution completed.  If dbcc printed error messages, see your System
Administrator.
```

Если для сегмента журнала отдельное устройство не выделено, будет выдано следующее сообщение:

```
*** NOTICE:  Notification of log space used/free cannot be reported because the
log segment is not on its own device.
```

dbcc tablealloc ({*имя_таблицы* | *идентификатор_таблицы*}

Имя или идентификатор таблицы, подлежащей проверке.

checkverify

Проверяет результаты последнего выполнения команды dbcc checkstorage для указанной базы данных. Подробную информацию о применении команды dbcc checkverify см. в книге *Руководство по системному администрированию*.

complete_xact

Эвристически завершает транзакцию путем ее фиксации или отката. Сервер Adaptive Server сохраняет информацию обо всех эвристически завершенных транзакциях в таблице master.dbo.systransactions, чтобы внешний координатор транзакций мог располагать некоторой информацией о завершении транзакций.

Предупреждение. Эвристическое завершение транзакции, находящейся в стадии подготовки, может привести к нарушению согласованности результатов всей распределенной транзакции. Решение системного администратора об эвристической фиксации или откате транзакции может вступить в противоречие с решением, принятым координирующим сервером Adaptive Server или протоколом.

---

forget_xact

Удаляет статус фиксации эвристически завершенной транзакции из таблицы master.dbo.systransactions. Параметр forget_xact может быть использован, когда системный администратор не хочет, чтобы служба координации знала об эвристическом завершении транзакции, либо когда статус фиксации не может быть удален из таблицы systransactions внешним координатором.

Предупреждение. Параметр dbcc forget_xact никогда не следует использовать в обычной среде обработки распределенных транзакций, поскольку внешнему координатору транзакций должно быть разрешено обнаруживать эвристически завершенные транзакции. Диспетчеры транзакций, совместимые со стандартом X/Open XA, и службы координации транзакций сервера Adaptive Server автоматически удаляют статус фиксации из таблицы systransactions.

идентификатор_транзакции

Имя транзакции из столбца systransactions.xactname. Допустимые значения параметра идентификатор_транзакции можно также определить при помощи системной процедуры sp_transactions.

dbrepair (имя_базы_данных, dropdb)

Удаляет поврежденную базу данных. Команда drop database не действует на поврежденные базы данных.

При выполнении этого оператора dbcc пользователи (в том числе и пользователь, выполняющий данный оператор) с удаляемой базой данных работать не могут.

fengine

Переводит экземпляры сервера Adaptive Server из оперативного режима в автономный и наоборот. Если *номер_экземпляра_сервера* не задан, то команда dbcc engine (offline) переводит в автономный режим экземпляр с наибольшим номером. Дополнительная информация содержится в главе 20 “Управление многопроцессорными серверами” книги *Руководство по системному администрированию*.

fix_text

Обновляет значения типа text после замены сервером Adaptive Server любого набора символов на новый многобайтовый набор символов.

Переход на многобайтовый набор символов усложняет внутреннее управление данными типа text. Поскольку значение типа text может быть достаточно большим и занимать несколько страниц, сервер

Adaptive Server должен уметь обрабатывать символы, часть байтов которых попала на одну страницу, а часть – на другую. Для этого серверу нужна дополнительная информация о каждой из страниц с данными типа text. Системный администратор или владелец таблицы должен выполнить команду dbcc fix_text для каждой таблицы, содержащей данные типа text, чтобы вычислить новые необходимые значения. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

indexalloc

Проверяет, что все страницы в указанном индексе выделены правильно и ни одна из выделенных страниц не осталась неиспользованной. Это сокращенная версия команды checkalloc, проверяющая целостность отдельного индекса по тем же критериям, что и полная версия.

При использовании параметра indexalloc будут сформированы те же три типа отчетов, что и при параметре tablealloc: full, optimized и fast. Если тип отчета не задан или указано значение NULL, сервер Adaptive Server использует тип optimized. Параметр fix | nofix работает с параметром indexalloc так же, как и с параметром tablealloc.

Примечание. Параметры fix или nofix можно задавать только вместе с указанием типа отчета (full, optimized, fast или null).

имя_таблицы | *идентификатор_таблицы*, *идентификатор_индекса*

Имя таблицы или объектный идентификатор таблицы (столбец id таблицы sysobjects), а также идентификатор индекса (столбец indid таблицы sysindexes).

full

Выдает отчет обо всех типах ошибок выделения пространства.

optimized

Выдает отчет на основе страниц распределения памяти, на которые указывают страницы карты размещения объекта (object allocation map, OAM). Выдает отчет по страницам распределения памяти, указанным в карте размещения объекта (object allocation map, OAM). Экстенты, указанные на страницах распределения памяти, на которые нет ссылок на страницах OAM, в отчет не входят и не могут быть исправлены. Установка optimized используется по умолчанию.

fast

Не выдает отчета о выделении памяти, но выдает отчет об исключениях, содержащий сведения о страницах, на которые есть ссылки, но которые не были выделены в экстенте (ошибки уровня 2521).

fix | nofix

Определяет, будет ли команда dbcc indexalloc устранять ошибки выделения памяти, обнаруженные в таблице. Параметр fix используется по умолчанию для всех индексов, кроме индексов системных таблиц, для которых по умолчанию применяется параметр nofix. Перед тем как применять параметр fix к системным таблицам, необходимо перевести базу данных в однопользовательский режим.

Параметры fix или nofix можно задавать только вместе с указанием типа отчета (full, optimized, fast или null).

rebuild_text

Перестраивает или создает внутреннюю структуру данных сервера Adaptive Server 12.x для данных типа text или image. Эта структура данных позволяет серверу Adaptive Server во время запросов осуществлять произвольный доступ к данным и выполнять асинхронную предварительную выборку.

reindex

Проверяет целостность индексов пользовательских таблиц при помощи “быстрой” версии команды dbcc checktable. Его можно указывать вместе с именем или объектным идентификатором таблицы (столбец id таблицы sysobjects). При обнаружении первой связанной с индексом ошибки команда dbcc reindex выдает соответствующее сообщение, а затем удаляет подозрительные индексы и заново их создает. Системный администратор или владелец таблицы должен выполнить команду dbcc reindex после того, как изменился порядок сортировки сервера Adaptive Server и индексы были помечены сервером как “подозрительные”.

Когда команда dbcc обнаруживает поврежденные индексы, она их удаляет и затем создает заново.Если индексы таблицы не повреждены или таблица не имеет индексов, команда dbcc reindex выдает соответствующее информационное сообщение, не перестраивая индексов.

Если возникает подозрение о повреждении данных в таблице, команда dbcc reindex аварийно завершается. Выдаваемое при этом сообщение об ошибке предписывает пользователю выполнить команду dbcc checktable. Команда dbcc reindex не позволяет перестраивать индексы системных таблиц. Системные индексы проверяются и при необходимости перестраиваются автоматически в процессе восстановления после перезапуска сервера Adaptive Server, следующего за изменением порядка сортировки.

tablealloc
Проверяет, чтобы все страницы в указанной таблице были выделены правильно и ни одна из выделенных страниц не осталась неиспользованной. Это сокращенная версия команды checkalloc, проверяющая целостность отдельной таблицы по тем же критериям, что и полная версия. Этот параметр можно указывать с именем или объектным идентификатором таблицы (столбец id таблицы sysobjects). Пример выходных данных команды dbcc tablealloc см. в книге *Руководство по системному администрированию*.

Команда tablealloc создает три типа отчетов: full, optimized и fast. Если тип отчета не задан или указано значение NULL, Adaptive Server использует тип optimized.

full
Эквивалентен параметру checkalloc на уровне таблиц; сообщает обо всех типах ошибок выделения памяти.

optimized
Выдает отчет на основе страниц выделения памяти, указанных на страницах OAM этой таблицы. Применение этого параметра не позволяет получать отчет или исправлять ошибки, связанные с отсутствием ссылок на экстенты, указанные на тех страницах выделения памяти, на которые нет указателей на страницах OAM. Установка optimized используется по умолчанию.

fast
Не выдает отчета о выделении памяти, но выдает отчет об исключениях, содержащий сведения о не выделенных в экстенте страницах, на которые есть ссылки (ошибки уровня 2521).

fix | nofix
Определяет, должна ли команда dbcc tablealloc устранять ошибки выделения памяти, обнаруженные в таблице. Параметр fix используется по умолчанию для всех таблиц, кроме системных таблиц, для которых по умолчанию применяется параметр nofix. Перед тем как применять параметр fix к системным таблицам, необходимо перевести базу данных в однопользовательский режим.

Параметры fix или nofix можно задавать только вместе с указанием типа отчета (full, optimized, fast или null).

traceon | traceoff
Включает или выключает печать диагностической информации во время оптимизации запросов (значения 302, 310 и 317 параметра *флаг*). Значения 3604 и 3605 включают или выключают пересылку выходных данных трассировки в сеанс пользователя и в журнал ошибок соот-

ветственно. Дополнительную информацию см. в главе 37 “Настройка при помощи команды dbcc traceon” книги *Руководство по настройке производительности*.

tune
: Включает или выключает флаги настройки в особых ситуациях при настройке производительности. Дополнительную информацию об отдельных параметрах см. в книге *Руководство по настройке производительности*.

Примеры

Пример 1. Проверка ошибок выделения страниц в базе данных pubs2:

```
dbcc checkalloc(pubs2)
```

Пример 2. Проверка согласованности базы данных pubs2 и помещение информации в базу данных dbccdb:

```
dbcc checkstorage(pubs2)
```

Пример 3. Сервер Adaptive Server возвращает отчет типа optimized о выделении страниц этой таблице, но не исправляет ошибки выделения:

```
dbcc tablealloc(publishers, null, nofix)
```

```
Checking salesdetail
The total number of pages in partition 1 is 3.
The total number of pages in partition 2 is 1.
The total number of pages in partition 3 is 1.
The total number of pages in partition 4 is 1.
The total number of data pages in this table is 10.
Table has 116 data rows.
DBCC execution completed. If DBCC printed error
messages, contact a user with System Administrator (SA)
role.
```

```
dbcc checktable(salesdetail)
```

Пример 4. Сервер Adaptive Server возвращает полный отчет (типа full) о страницах индекса по таблице titleauthor с идентификатором indid, равным 2, и исправляет все ошибки выделения памяти:

```
dbcc indexalloc ("pubs..titleauthor", 2, full)
```

Пример 5. Перестройка или создание внутренней структуры данных сервера Adaptive Server 12.x для всех столбцов типа text или image таблицы blurbs:

```
dbcc rebuild_text (blurbs)
```

Пример 6. Команда dbcc reindex обнаружила один или несколько поврежденных индексов таблицы titles:

```
dbcc reindex(titles)
```

```
One or more indexes are corrupt. They will be rebuilt.
```

Пример 7. Обновление текстовых значений таблицы blurbs после смены набора символов:

```
dbcc fix_text(blurbs)
```

Пример 8. Эвристическое завершение транзакции “distributedxact1”:

```
dbcc complete_xact (distributedxact1, "rollback")
```

Пример 9. Удаление информации о транзакции “distributedxact1” из таблицы master.dbo.systransactions:

```
dbcc forget_xact (distributedxact1)
```

Использование

- Команду dbcc (Database Consistency Checker) можно выполнять, когда база данных активна, за исключением параметра dbrepair(имя_базы_данных, dropdb) и команды dbcc checkalloc с параметром fix.
- Команда dbcc блокирует объекты базы данных в процессе их проверки. Рекомендации по минимизации влияния команды dbcc на производительность приводятся в описании команды dbcc в книге *Руководство по системному администрированию*.
- При уточнении имени таблицы или индекса именем пользователя или базы данных необходимо заключать уточненное имя в одинарные или двойные кавычки. Например:

  ```
  dbcc tablealloc("pubs2.pogo.testtable")
  ```

- Команду dbcc reindex нельзя выполнять внутри пользовательской транзакции.
- Команда dbcc fix_text способна создавать большое количество журнальных записей, которые могут полностью заполнять журнал транзакций. При проектировании команды dbcc fix_text было предусмотрено, чтобы обновления выполнялись в виде последовательности небольших транзакций и при отсутствии места в журнале терялась бы лишь небольшая часть работы. Если место в журнале закончится, нужно очистить журнал и снова выполнить команду dbcc fix_text для таблицы, во время обновления которой произошла ошибка в исходной команде dbcc fix_text.
- После перехода на многобайтовый набор символов необходимо выполнить команду dbcc fix_text. Если этого не сделать, то попытка прочитать или записать значения типа text с помощью оператора select, readtext или writetext приведет к ошибке, и будет выведено сообщение с требованием выполнить команду dbcc fix_text для данной таблицы. Однако удалять строки типа text после изменения наборов символов можно без выполнения команды dbcc fix_text.

- Выходные данные команды dbcc пересылаются как информационные сообщения или сообщения об ошибках, а не в виде результирующих строк. Клиентские программы и сценарии должны проверять соответствующие обработчики ошибок.
- Если таблица секционирована, команда dbcc checktable возвращает информацию о каждой секции.
- Данные типа text и image, которые были перенесены на сервер Adaptive Server версии 12.x, *не* преобразуются автоматически в новый формат хранения. Чтобы повысить производительность запросов и обеспечить возможность предварительной выборки этих данных, для перенесенных столбцов типа text и image используется ключевое слово rebuild_text.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду dbcc с ключевыми словами checktable, fix_text, rebuild_text или reindex может выполнять только владелец таблицы.

Ключевые слова checkstorage, checkdb, checkcatalog, checkalloc, indexalloc и tablealloc может использовать только владелец базы данных.

Ключевые слова dbrepair, complete_xact и forget_xact может использовать только системный администратор.

Команды dbcc traceon и dbcc traceoff может выполнять только системный администратор.

Команду dbcc engine может выполнять только системный администратор.

См. также

Команды drop database

Системные процедуры sp_configure, sp_helpdb

deallocate cursor

Описание

Делает курсор недоступным и освобождает все ресурсы памяти, выделенные этому курсору.

Синтаксис

deallocate cursor *имя_курсора*

Параметры

имя_курсора
Имя курсора, подлежащего удалению.

Примеры

Удаление курсора “authors_crsr”:

```
deallocate cursor authors_crsr
```

Использование

- Если курсор не существует, будет возвращено сообщение об ошибке.
- Нужно удалить курсор, перед тем как его имя можно использовать в новом операторе declare cursor.
- Команда deallocate cursor, указанная в хранимой процедуре или триггере, не влияет на использование ресурсов памяти.
- Команду deallocate можно выполнять как над открытым, так и над закрытым курсором.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду deallocate cursor по умолчанию могут выполнять все пользователи. Для выполнения этой команды не требуется обладать какими-либо полномочиями.

См. также

Команды close, declare cursor

declare

Описание

Объявляет имена и типы локальных переменных для пакета или процедуры.

Синтаксис

Объявление переменной:

declare @*имя_переменной тип_данных*
[, @*имя_переменной тип_данных*]...

Присвоение значения переменной:

select @*переменная* = {*выражение* | *оператор_select*}
[, @*переменная* = {*выражение* | *оператор_select*} ...]
[from *список_таблиц*]
[where *условия_поиска*]
[group by *список_group_by*]
[having *условия_поиска*]
[order by *список_order_by*]
[compute *список_функций* [by *список_столбцов*]]

Параметры

@имя_переменной

Должно начинаться со знака @ и соответствовать правилам для идентификаторов.

тип_данных

Системным тип данных или тип данных, определенный пользователем.

Примеры

Пример 1. Объявление двух переменных и печать строк в зависимости от значений переменных.

```
declare @one varchar(18), @two varchar(18)
select @one = "this is one", @two = "this is two"
if @one = "this is one"
    print "you got one"
if @two = "this is two"
    print "you got two"
else print "nope"
```

```
you got one
you got two
```

Пример 2. Печать строки “Ouch!”, если максимальная цена книги в таблице titles больше $20.00:

```
declare @veryhigh money
select @veryhigh = max(price)
    from titles
if @veryhigh > $20
    print "Ouch!"
```

Использование

- Значения локальным переменным присваиваются при помощи оператора select.
- Максимальное число параметров в процедуре равно 2048. Число локальных или глобальных переменных ограничивается только объемом доступной памяти. Знак @ обозначает имя переменной.
- Локальные переменные часто используются в качестве счетчиков в циклах while и блоках if...else. В хранимых процедурах они объявляются для автоматического, неинтерактивного использования процедурой в процессе ее выполнения. Локальные переменные должны использоваться только в том пакете или процедуре, в которых они объявлены.
- Оператор select, посредством которого присваивается значение локальной переменной, обычно возвращает единственное значение. Если должно возвращаться более одного значения, переменной присваивается последнее из них. Оператор select, который присваивает значения переменным, не может одновременно с этим использоваться для выборки данных.
- Команды print и raiserror могут использовать локальные переменные в качестве аргументов.
- Пользователи не могут создавать глобальные переменные и не могут изменять их значения непосредственно в операторе select.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду declare по умолчанию могут выполнять все пользователи. Для выполнения этой команды не требуется обладать какими-либо полномочиями.

См. также

Команды print, raiserror, select, while

declare cursor

Описание

Определяет курсор.

Синтаксис

declare *имя_курсора* cursor
for *оператор_select*
[for {read only | update [of *список_имен_столбцов*]}]

Параметры

имя_курсора
Имя определяемого курсора.

оператор_select
Запрос, который определяет результирующий набор курсора. Дополнительную информацию см. в описании оператора select.

for read only
Указывает, что результирующий набор курсора не подлежит обновлению.

for update
Указывает, что результирующий набор курсора можно обновлять.

of список_имен_столбцов
Список столбцов из обновляемого результирующего набора курсора (определенного оператором *оператор_select*). Сервер Adaptive Server позволяет в список обновляемых столбцов включать также столбцы, принадлежащие таблицам, упомянутым в операторе *оператор_select*, но не указанным в списке столбцов этого оператора (и следовательно, не попавшим в результирующий набор).

Примеры

Пример 1. Определение результирующего набора для курсора authors_crsr, который содержит идентификатор, имя и фамилию всех авторов из таблицы authors, которые живут не в Калифорнии:

```
declare authors_crsr cursor
for select au_id, au_lname, au_fname
from authors
where state != 'CA'
```

Пример 2. Определение доступного только для чтения результирующего набора для курсора titles_crsr, который содержит информацию о книгах по бизнесу из таблицы titles:

```
declare titles_crsr cursor
for select title, title_id from titles
where title_id like "BU%"
for read only
```

Пример 3. Определение обновляемого результирующего набора для курсора pubs_crsr, который содержит все строки таблицы publishers. В этом результирующем наборе обновляемыми являются адреса издателей (столбцы city и state):

```
declare pubs_crsr cursor
for select pub_name, city, state
from publishers
for update of city, state
```

Использование

Ограничения, накладываемые на курсоры

- Оператор declare cursor должен предшествовать каждому оператору openдля этого курсора.
- Оператор declare cursor нельзя использовать совместно с другими операторами в одном пакете Transact-SQL.
- В инструкцию update оператора declare cursor на стороне клиента можно включать до 1024 столбцов.
- *имя_курсора* должно быть допустимым идентификатором сервера Adaptive Server.

Операторы *select* для курсора

- В *операторе_select* можно в полной мере использовать синтаксис и семантику оператора select пакета Transact-SQL с учетом следующих ограничений:
  - инструкция from является обязательной;
  - нельзя использовать инструкции compute, for browse и into;
  - можно использовать ключевое слово holdlock.
- *Оператор_select* может содержать ссылки на имена параметров или локальные переменные пакета Transact-SQL (для всех типов курсоров, кроме языковых). Эти параметры и локальные переменные должны быть определены в той процедуре, триггере или пакете операторов, где содержится оператор declare cursor.

  Параметры и локальные переменные, указанные в операторе declare cursor, могут не иметь допустимых значений, пока курсор не открыт.
- *Оператор_select* может содержать ссылки на временные таблицы inserted и deleted, которые используются в триггерах.

Область действия курсора

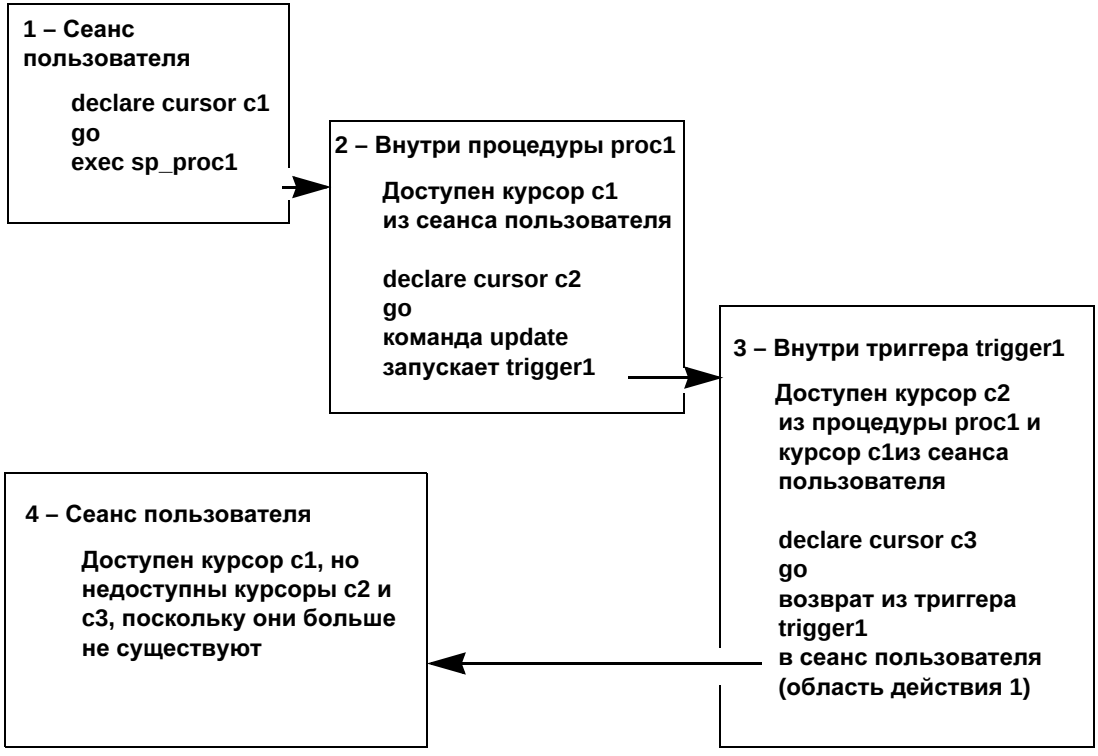
- Длительность существования курсора зависит от его *области действия*. Область действия – это контекст, в котором используется курсор (сеанс пользователя, хранимая процедура или триггер).

В сеансе пользователя курсор существует до тех пор, пока пользователь не закончит сеанс. Курсор не будет существовать в других сеансах, установленных другими пользователями. После выхода пользователя из системы сервер Adaptive Server удаляет все курсоры, созданные в сеансе данного пользователя.

Если оператор declare cursor является частью хранимой процедуры или триггера, то созданный этим оператором курсор относится к области действия хранимой процедуры или триггера, а также к области действия, из которой были запущены эта процедура или триггер. Курсоры, объявленные внутри триггера для таблицы inserted или deleted, доступны в пределах области действия данного триггера, но недоступны для всех вложенных триггеров и хранимых процедур. После выполнения процедуры или триггера сервер Adaptive Server удаляет все созданные в них курсоры.

Рис. 7-1 иллюстрирует доступность курсоров в различных областях действия.

Рис. 7-1. Доступность курсоров в различных областях действия



- Имя курсора должно быть уникальным в данной области действия. Сервер Adaptive Server обнаруживает конфликты имен в конкретной области действия только во время выполнения. В хранимой процедуре или триггере можно определять два курсора с одинаковым именем при условии, что выполняться будет лишь один из них. Например, следующая хранимая процедура будет работать, потому что в ее области действия определен лишь один курсор names_crsr:

```
create procedure proc2 @flag int
as
if @flag > 0
    declare names_crsr cursor
    for select au_fname from authors
else
    declare names_crsr cursor
    for select au_lname from authors
return
```

Результирующий набор

- Строки в результирующем наборе курсора могут не отражать фактические значения в строках базовой таблицы. Например, для курсора, объявленного с использованием инструкции order by, обычно требуется создание внутренней таблицы для упорядочивания строк результирующего набора. Сервер Adaptive Server не блокирует строки базовой таблицы, соответствующие строкам внутренней таблицы, что позволяет другим клиентам обновлять эти строки в базовой таблице. В этом случае строки, возвращаемые клиенту из результирующего набора курсора, не будут синхронизированы со строками базовой таблицы.

- Результирующий набор курсора создается по мере возврата строк посредством оператора fetch для этого курсора. Это означает, что запрос select к курсору обрабатывается так же, как обычный запрос select. Этот процесс, известный под названием *сканирование курсора*, обеспечивает сокращение времени выполнения запроса и устраняет необходимость чтения строк, которые не требуются приложению.

  Сканирование курсора ограничено тем, что оно может использовать только уникальные индексы таблицы. Однако если ни одна из базовых таблиц, на которые есть ссылки в результирующем наборе курсора, не обновляется другим процессом в том же пространстве блокировок, что и курсор, это ограничение не является обязательным. Сервер Adaptive Server позволяет объявлять курсоры для таблиц, не имеющих уникальных индексов, однако любая попытка обновления этих таблиц в том же самом пространстве блокировок приводит к закрытию всех курсоров для таблиц.

Обновляемые курсоры

- После определения курсора при помощи оператора declare cursor сервер Adaptive Server выясняет, является ли курсор *обновляемым* или доступным *только для чтения*. Если курсор обновляемый, в его результирующем наборе можно обновлять или удалять строки. Если курсор доступен только для чтения, изменять результирующий набор нельзя.
- Инструкции for update или for read only используются, чтобы явно определять, является ли курсор обновляемым или доступным только для чтения. Нельзя объявить курсор как обновляемый, если его *оператор_select* содержит одну из следующих конструкций:
  - параметр distinct;
  - инструкцию group by;
  - агрегатную функцию;
  - подзапрос;
  - оператор union ;
  - инструкцию at isolation read uncommitted.

  Если инструкция for update или read only пропущена, то сервер Adaptive Server проверяет, является ли курсор обновляемым.

  Языковые курсоры и курсоры сервера, чей *оператор_select* включает инструкцию order by, доступны только для чтения. Adaptive Server по-разному обрабатывает обновления курсоров клиента и курсоров выполнения, устраняя таким образом это ограничение.
- Если в инструкции for update не задан параметр *список_имен_столбцов*, все указанные в запросе столбцы являются обновляемыми. Сервер Adaptive Server пытается использовать уникальные индексы для обновляемых курсоров при сканировании базовой таблицы. В применении к курсорам Adaptive Server рассматривает индекс, содержащий столбец IDENTITY, как уникальный, даже если уникальность индекса явным образом не объявлена.

  Если инструкция for update не задана, сервер Adaptive Server выбирает любой уникальный индекс, хотя он может использовать также другие индексы для сканирования таблиц, если не существует уникального индекса для заданных столбцов таблицы. Однако при наличии инструкции for update Adaptive Server должен для сканирования базовой таблицы использовать уникальный индекс, определенный для одного или нескольких столбцов. Если такового не существует, сервер возвращает сообщение об ошибке.

- В большинстве случаев в *список_имен_столбцов* инструкции for update включаются только те столбцы, значения которых будут изменены. Если для таблицы существует лишь один уникальный индекс, его столбец не обязательно включать в инструкцию for update список_имен_столбцов; сервер Adaptive Server сам найдет его при сканировании курсора. Если для таблицы существует более одного уникального индекса, включите столбец одного из них в инструкцию for update список_имен_столбцов, чтобы Adaptive Server смог быстрее найти его для сканирования курсора.

  Это позволяет серверу Adaptive Server использовать данный уникальный индекс при сканировании курсора, что помогает предотвращать аномалию, называемую **проблемой Хеллоуина**. Другой способ предотвращения проблемы Хеллоуина заключается в создании таблиц с параметром базы данных unique auto_identity index. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

  Проблема Хеллоуина возникает в том случае, когда клиент в строке результирующего набора курсора обновляет столбец, определяющий порядок, в котором строки выбираются из базовой таблицы. Например, если сервер Adaptive Server обращается к базовой таблице с использованием индекса и ключ индекса обновляется клиентом, обновленная строка индекса может переместиться в другое место индекса и вновь быть считана курсором. Такая возможность является следствием того, что результирующий набор обновляемого курсора создается лишь как логическая структура данных. Фактически результирующим набором являются сами базовые таблицы, для которых определен курсор.

- Если задан параметр read only, результирующий набор курсора нельзя обновлять при помощи операторов delete или update.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Параметры for update и for read only являются расширениями Transact-SQL.

Полномочия

Команду declare cursor по умолчанию могут выполнять все пользователи. Для выполнения этой команды не требуется обладать какими-либо полномочиями.

См. также

Команда open

delete

Описание

Удаляет строки из таблицы.

Синтаксис

```
delete [from]
        [[база_данных.]владелец.]{имя_представления|имя_таблицы}
        [where условия_поиска]
        [plan "абстрактный план"]

delete [[база_данных.]владелец.]{имя_таблицы | имя_представления}
        [from [[база_данных.]владелец.]{имя_представления [readpast]|
            имя_таблицы [readpast]
                [(index {имя_индекса | имя_таблицы }
                [ prefetch размер ][lru|mru])]}
        [, [[база_данных.]владелец.]{имя_представления [readpast]|
            имя_таблицы [readpast]
                [(index {имя_индекса | имя_таблицы }
                [ размер предварительной выборки ][lru|mru])]} ...]
        [where условия_поиска] ]
        [plan "абстрактный план"]

delete [from]
[[база_данных.]владелец.]{имя_таблицы|имя_представления}
        where current of имя_курсора
```

Параметры

from (после delete)

Необязательное ключевое слово, используемое для обеспечения совместимости с другими версиями SQL.

имя_представления | *имя_таблицы*

Имя представления или таблицы, из которых необходимо удалить строки. Если таблица или представление находится в другой базе данных, то необходимо указать имя базы данных, а если в базе данных существует несколько таблиц или представлений с этим именем – имя их владельца. По умолчанию *владелец* – это текущий пользователь, а *база_данных* – текущая база данных.

where

Стандартная инструкция where. Дополнительную информацию см. в разделе where.

from (после *имя_таблицы* или *имя_представления*)

Позволяет указывать несколько таблиц или представлений для использования с инструкцией where при указании строк, которые необходимо удалить. Эта инструкция from позволяет удалять строки из одной таблицы на основе данных, хранимых в других таблицах, обеспечивая тем самым значительную часть возможностей вложенного оператора select.

readpast

Указывает, что команда delete будет пропускать все страницы или строки с несовместимыми блокировками, не ожидая снятия этих блокировок. В таблицах с блокировками страниц данных пропускаются все строки, расположенные на страницах с несовместимыми блокировками; в таблицах с блокировками строк данных пропускаются все строки с несовместимыми блокировками.

index имя_индекса

Определяет индекс, который будет использоваться для доступа к таблице *имя_таблицы*. При удалении строк из представления этот параметр использовать нельзя.

prefetch размер

Задает размер блока ввода-вывода в килобайтах для таблиц, связанных с кэшами для операций ввода-вывода большими блоками. Этот параметр нельзя использовать при удалении строк из представления. Размеры буферов для кэша, к которому прикреплен объект (или для кэша по умолчанию, если объект не прикреплен ни к одному кэшу) можно узнать с помощью системной процедуры sp_helpcache.

Если указано ключевое слово prefetch, то допустимыми значениями размера предварительной выборки (*размер*) являются произведения размера логической страницы (от 2 до 16 КБ) на числа 1,2,4 и 8. Эти допустимые значения размера предварительной выборки приведены в следующей таблице (в килобайтах):

Размер логической страницы	Размеры предварительной выборки
2	2, 4, 8 16
4	4, 8, 16, 32
8	8, 16, 32, 64
16	16, 32, 64, 128

Указанное в запросе значение размера предварительной выборки носит лишь рекомендательный характер. Чтобы указанный размер можно было использовать, этот размер буферов необходимо сконфигурировать в кэше данных. Если этого не сделать, будет использоваться значение параметра prefetch, заданное по умолчанию.

Для конфигурирования размеров кэша данных используется системная процедура sp_cacheconfigure.

Примечание. Если активны службы Component Integration Services, ключевое слово prefetch нельзя использовать для удаленных серверов.

lru | mru
: Задает стратегию замены буферов для таблицы. Параметр lru предписывает оптимизатору считывать таблицу в кэш по цепочке MRU/LRU (most recently used/least recently used – недавно/давно использовавшийся). Параметр mru используется для удаления буфера из кэша и замены его следующим буфером для таблицы. При удалении строк из представления этот параметр использовать нельзя.

plan "абстрактный план"
: Задает абстрактный план для оптимизации запроса. Это может быть полный или частичный план, заданный на языке абстрактных планов. Дополнительную информацию см. в главе 22 “Создание и использование абстрактных планов” книги *Руководство по настройке производительности*.

where current of имя_курсора
: Удаляет строку таблицы или представления, находящуюся в текущей позиции курсора *имя_курсора*.

Примеры

Пример 1. Удаление всех строк из таблицы authors:

```
delete authors
```

Пример 2. Удаление одной или нескольких строк из таблицы authors:

```
delete from authors
where au_lname = "McBadden"
```

Пример 3. Удаление строк, соответствующих книгам, написанным автором по фамилии Bennet, из таблицы titles:

```
delete titles
from titles, authors, titleauthor
where authors.au_lname = 'Bennet'
  and authors.au_id = titleauthor.au_id
  and titleauthor.title_id = titles.title_id
```

В базе данных pubs2 имеется триггер (deltitle), который предотвращает удаление названий, занесенных в таблицу sales; для правильной работы примера этот триггер необходимо удалить.

Пример 4. Удаление из таблицы titles строки, на которую в данный момент указывает курсор title_crsr:

```
delete titles where current of title_crsr
```

Пример 5. Нахождение строки, которая в столбце IDENTITY имеет значение 4, и удаление этой строки из таблицы authors. Обратите внимание на то, что вместо фактического имени столбца IDENTITY используется ключевое слово syb_identity:

```
delete authors
where syb_identity = 4
```

Пример 6. Удаление строк из таблицы authors с пропуском всех заблокированных строк:

```
delete from authors from authors readpast
where state = "CA"
```

Пример 7. Удаление строк из таблицы stores с пропуском всех заблокированных строк. Если какие-либо строки в таблице authors заблокированы, запрос останавливается на этих строках в ожидании снятия блокировок:

```
delete stores from stores readpast, authors
where stores.city = authors.city
```

Использование

- Оператор delete удаляет строки из указанной таблицы.
- В операторе delete можно указывать до 15 таблиц (включительно).

Ограничения

- Команду delete нельзя использовать для многотабличного представления (т.е. когда инструкция from содержит более одной таблицы представления), даже если к этому представлению можно применять команды update или insert. Удаление строки из многотабличного представления изменяет сразу несколько таблиц, что недопустимо. Команды insert и update разрешается применять только в том случае, когда они затрагивают лишь одну из базовых таблиц.
- Adaptive Server рассматривает два разных обозначения одной и той же таблицы в команде delete как две таблицы. Например, следующая команда delete, выполняемая в базе данных pubs2, определяет таблицу discounts как две таблицы (discounts и pubs2..discounts):

  ```
  delete discounts
  from pubs2..discounts, pubs2..stores
  where pubs2..discounts.stor_id =
     pubs2..stores.stor_id
  ```

  В этом случае соединение не включает таблицу discounts, поэтому условие where остается истинным для каждой строки; Adaptive Server удаляет все строки в таблице discounts (что не является желаемым результатом). Во избежание подобных проблем следует использовать одно и то же обозначение таблицы на протяжении всего оператора.

- Если из таблицы, являющейся родительской в ограничении ссылочной целостности, удаляется строка, то Adaptive Server проверяет дочерние таблицы, ссылающиеся на эту таблицу, перед тем как разрешить удаление строки. Если удаляемая строка содержит первичный ключ, на который ссылается внешний ключ в какой-либо дочерней таблице, удаление не разрешается.

Удаление всех строк из таблицы

- Если инструкция where не задана, удаляются *все* строки из таблицы, указанной после команды delete [from]. Пустая таблица остается в базе данных до тех пор, пока для нее не будет выполнена команда drop table.
- Команды truncate table и delete без указания строк функционально эквивалентны, но оператор truncate table работает быстрее. Команда delete удаляет строки по одной, занося эти транзакции в журнал. Оператор truncate table удаляет страницы данных целиком и не отражает строки в журнале.

  Оба оператора (и delete, и truncate table) освобождают пространство, занятое данными и связанными с ними индексами.
- Команду truncate table нельзя использовать для секционированной таблицы. Для удаления всех строк из секционированной таблицы нужно либо использовать команду delete без инструкции where, либо десекционировать таблицу перед применением команды truncate table.

Команда *delete* и транзакции

- В режиме связанных транзакций каждый оператор delete неявным образом начинает новую транзакцию, если в данный момент нет текущей активной транзакции. Для завершения всех операций удаления используется оператор commit, а для отката изменений – оператор rollback. Например:

```
delete from sales where date < '01/01/89'
if exists (select stor_id
   from stores
   where stor_id not in
   (select stor_id from sales))
      rollback transaction
else
      commit transaction
```

Этот пакет начинает транзакцию (в режиме связанных транзакций) и удаляет из таблицы sales строки с датами, предшествующими 1 января 1989 г. Если удаляются все записи о продажах (sales), связанные с конкретным магазином (store), данный пакет производит откат всех изменений в таблице sales и завершает транзакцию. В противном случае он фиксирует операции удаления и завершает транзакцию. Дополнительную информацию о режиме связанных транзакций см. в книге *Руководство пользователя Transact-SQL*.

Триггеры команды *delete*

- Для выполнения заранее предусмотренных действий при обращении команды delete к заданной таблице можно определять триггер.

Использование команды *delete where current of*

- Инструкция where current of используется с курсорами. Перед удалением строк с использованием инструкции where current of необходимо сначала при помощи команды declare cursor определить курсор и открыть его оператором open. Затем курсор устанавливается на строку, подлежащую удалению, при помощи одного или нескольких операторов fetch. Имя курсора не может быть параметром Transact-SQL или локальной переменной. Курсор должен быть обновляемым, иначе сервер Adaptive Server выдаст сообщение об ошибке. Удаление любой строки из результирующего набора курсора приводит к удалению строки в базовой таблице, из которой получена строка курсора. При помощи курсора строки можно удалять только по одной.

- Даже если курсор считается обновляемым, удалять строки из его результирующего набора нельзя, когда оператор select для курсора содержит инструкцию соединения. Переменные *имя_таблицы* или *имя_представления* в операторе delete...where current of должны совпадать с именами таблицы или представления, указанными в первой инструкции from оператора select, определяющего курсор.

- После удаления строки из результирующего набора курсор располагается перед следующей строкой в наборе. Для получения доступа к следующей строке нужно выполнить оператор fetch. Если удаленная строка была последней в результирующем наборе курсора, курсор располагается за последней строкой результирующего набора. Ниже поясняется позиционирование и поведение открытых курсоров, затронутых оператором delete.

    - Если с помощью курсора или обычного оператора delete клиент удаляет строку, соответствующую текущей позиции других открытых курсоров, принадлежащих этому клиенту, каждый затронутый курсор неявным образом располагается перед следующей

доступной строкой. Однако клиент не может удалить строку, соответствующую текущей позиции курсора, принадлежащего другому клиенту.

- Если клиент удаляет строку, соответствующую текущей позиции другого курсора, определенного с помощью операции соединения и принадлежащего этому же клиенту, сервер Adaptive Server разрешит выполнить оператор delete. Однако при этом он неявным образом закроет курсор, определенный операцией соединения.

Использование параметра *readpast*

- Параметр readpast позволяет командам delete для таблиц, в которых заблокированы только данные, продолжать выполнение, не ожидая снятия несовместимых блокировок, удерживаемых другими задачами.
  - В таблицах с блокировкой строк данных параметр readpast позволяет пропускать все строки, на которых другая задача удерживает монопольные блокировки, разделяемые блокировки или блокировки обновления.
  - В таблицах с блокировкой страниц данных параметр readpast позволяет пропускать все страницы, на которых другая задача удерживает монопольные блокировки, разделяемые блокировки или блокировки обновления.
- Команды с параметром readpast останавливаются, если имеет место монопольная блокировка таблицы.
- Если параметр readpast указан для таблицы с блокировкой всех страниц, параметр readpast игнорируется. Команда останавливается, как только она обнаруживает несовместимую блокировку.
- Если в сеансе уровень изоляции равен 3, параметр readpast игнорируется без уведомления. Команда выполняется на уровне 3. Она останавливается на любых строках или страницах с несовместимыми блокировками.
- Если в сеансе уровень изоляции транзакций равен 0, команда delete с параметром readpast не выдает предупреждающих сообщений. В таблицах с блокировкой страниц данных команда delete с параметром readpast удаляет все строки на всех страницах, не заблокированных несовместимыми блокировками. В таблицах с блокировкой строк данных он затрагивает все строки, не заблокированные несовместимыми блокировками.

- Если команда delete применяется к строке, включающей два или более текстовых столбца, и на любом из этих столбцов удерживается несовместимая блокировка, параметр readpast пропускает эту строку.

Использование параметров *index*, *prefetch* и *lru | mru*

- Параметры index, prefetch и lru | mru переопределяют значения, выбранные оптимизатором Adaptive Server. Эти параметры следует использовать с осторожностью и всегда проверять их влияние на производительность при помощи команды set statistics io on. Дополнительную информацию об использовании этих параметров см. в книге *Руководство по настройке производительности*.

Стандарты

Уроввень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Использование нескольких таблиц в инструкции from и уточнение имени таблицы с помощью имени базы данных являются расширениями Transact-SQL.

Параметр readpast является расширением Transact-SQL.

Полномочия

Полномочиями на выполнение команды delete по умолчанию обладает владелец таблицы или представления, который может передать эти полномочия другим пользователям.

Если параметр ansi_permissions установлен в on, то в дополнение к обычным полномочиям, необходимым для использования операторов delete, нужно иметь полномочия select на все столбцы, указанные в инструкции where. По умолчанию параметр ansi_permissions равен off.

См. также

Команды create trigger, drop table, drop trigger, truncate table, where

delete statistics

Описание

Удаляет статистику из системной таблицы sysstatistics.

Синтаксис

delete [shared] statistics *имя_таблицы* [(*имя_столбца* [, *имя_столбца*]...)]

Параметры

shared

Удаляет статистическую информацию, полученную по результатам моделирования, из таблицы sysstatistics базы данных master.

имя_таблицы

Удаляет статистику всех столбцов указанной таблицы.

имя_столбца

Удаляет статистику для указанного столбца.

Примеры

Пример 1. Удаление показателей плотности и селективности, а также гистограмм для всех столбцов таблицы titles:

```
delete statistics titles
```

Пример 2. Удаление показателей плотности и селективности, а также гистограмм для столбца pub_id таблицы titles:

```
delete statistics titles(pub_id)
```

Пример 3. Удаление показателей плотности и селективности, а также гистограмм для пары столбцов pub_id, pubdate с сохранением индивидуальных статистик столбцов pub_id и pubdate:

```
delete statistics titles(pub_id, pubdate)
```

Использование

- Команда delete statistics удаляет статистику указанной таблицы или указанных столбцов из таблицы sysstatistics. Она не затрагивает статистические данные в таблице systabstats table.
- При удалении таблицы с помощью команды drop table удаляются также соответствующие строки в таблице sysstatistics. При удалении индекса с помощью команды drop index строки в таблице sysstatistics не удаляются. Это позволяет оптимизатору запросов продолжать использование статистики индекса без издержек на ведение индекса таблицы.

Предупреждение. Показатели плотности и селективности, а также гистограммы необходимы для качественной оптимизации запросов. Команда delete statistics предназначена для удаления статистики, не используемой оптимизатором. Если вы случайно удалили статистику, необходимую для оптимизации запросов, выполните команду update statistics для таблицы, индекса или столбца.

- Загрузка статистической информации, полученной по результатам моделирования, с помощью утилиты optdiag добавляет небольшое количество строк в таблицу master..sysstatistics. Если статистика по результатам моделирования больше не используется, информацию из таблицы master..sysstatistics можно удалить при помощи команды delete shared statistics.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду delete statistics может выполнять только владелец таблицы и системный администратор.

См. также

Команды create index, update

Утилиты optdiag

disk init

Описание

Делает физическое устройство или файл доступным для СУБД Adaptive Server.

Синтаксис

```
disk init
        name = "имя_устройства" ,
        physname = "размещение_устройства" ,
        [vdevno = номер_виртуального_устройства ,]
        size = число_блоков
        [, vstart = виртуальный_адрес
             , cntrltype = номер_контроллера ]
        [, contiguous]
        [, dsync = { true | false } ]
```

Параметры

name

Имя устройства хранения базы данных или файла. Должно соответствовать правилам именования идентификаторов и должно быть заключено в одинарные или двойные кавычки. Это имя используется в командах create database и alter database.

physname

Полная спецификация устройства хранения базы данных. Должна быть заключена в одинарные или двойные кавычки.

vdevno

Номер виртуального устройства, который должен быть уникальным для любого устройства базы данных, связанного с Adaptive Server. Номер устройства 0 зарезервирован для главного устройства. Можно использовать номера в диапазоне от 1 до 255, но самый большой номер должен быть на единицу меньше общего числа устройств хранения базы данных, на работу с которыми настроен сервер Adaptive Server. Например, если сервер Adaptive Server по умолчанию настроен на работу с 10 устройствами, то должны использоваться номера устройств в диапазоне от 1 до 9. Чтобы узнать максимальное число устройств, доступных серверу Adaptive Server, выполните процедуру sp_configure и посмотрите, какое значение будет выведено для параметра number of devices.

Чтобы определить номер виртуального устройства, нужно найти в отчете процедуры sp_helpdevice столбец device_number и использовать следующий незадействованный номер.

size

Размер области, которая выделяется при расширении базы данных. Размер может быть задан в следующих единицах: ‘k’ или ‘K’ (килобайты), ‘м’ или ‘M’ (мегабайты) и ‘g’ или ‘G’ (гигабайты). Рекоменду-

ется всегда указывать единицы измерения. Если единица измерения не указана, кавычки можно опустить. Однако если она указана, кавычки нужны.

vstart

Начальный виртуальный адрес (смещение), начиная с которого сервер Adaptive Server будет использовать пространство устройства хранения базы данных. В параметре vstart можно указывать следующие обозначения единиц измерения пространства: 'k' или 'K' (килобайты), 'м' или 'M' (мегабайты) и 'g' или 'G' (гигабайты). Величина смещения зависит от того, каким образом введено значение параметра vstart.

- Если единица измерения не указана, для вычисления начального адреса используются страницы размером 2 КБ. Например, если задано `vstart = 13`, то сервер Adaptive Server использует в качестве смещения для начального адреса величину, равную 13 * 2 КБ.
- Если единица измерения указана, то в качестве начального адреса используется заданное значение. Например, если задано `vstart = "13M"`, сервер Adaptive Server устанавливает для начального адреса смещение, равное 13 мегабайтам.

Значение по умолчанию (и, как правило, предпочтительное значение) для параметра vstart равно 0. Если на заданном устройстве количество доступных блоков меньше суммы значений vstart + size, выполнение команды disk init заканчивается неудачей. При работе в операционной системе AIX и использовании утилиты Logical Volume Manager значение параметра vstart должно равняться 2. Задавать параметру vstart другие значения можно только в том случае, если такое указание было дано службой технической поддержки Sybase.

cntrltype

Контроллер диска. Значение по умолчанию равно 0. Изменять значение параметра cntrltype можно только в том случае, если такое указание было дано службой технической поддержки Sybase.

dsync

Только для платформ UNIX. Если используются файлы операционной системы UNIX, этот параметр определяет, будут ли операции записи на устройство хранения базы данных записывать информацию прямо на носитель или использовать буферизацию. Использовать этот параметр имеет смысл только в том случае, если инициализируемый файл находится в операционной системе UNIX. Он не оказывает никакого действия при инициализации устройств в неотформатированных (raw) разделах. По умолчанию при инициализации всех файлов операционной системы UNIX параметр dsync имеет значение true.

Примеры

Пример 1. Инициализация 5 МБ дискового пространства в системе UNIX.

```
disk init
name = "user_disk",
physname = "/dev/rxy1a",
vdevno = 2, size = 5120
```

Пример 2. Инициализация 10 МБ дискового пространства в файле операционной системы UNIX. СУБД Adaptive Server открывает файл, используя заданное значение параметра dsync, и при записи в файл данные гарантированно записываются прямо на носитель:

```
disk init
name = "user_file",
physname = "/usr/u/sybase/data/userfile1.dat",
vdevno = 2, size = 5120, dsync = true
```

Использование

- Главное устройство инициализируется программой установки, для его инициализации команда disk init не применяется.
- Чтобы инициализация диска прошла успешно, пользователю “sybase” на уровне операционной системы должны быть предоставлены соответствующие полномочия на инициализируемое устройство.
- Размер можно задавать в виде числа с плавающей точкой, но оно будет округлено в сторону уменьшения до ближайшего числа, кратного 2 КБ.
- Если не указана единица измерения для параметра size:
  - Команда disk init использует виртуальные страницы размером 2 КБ.
  - Значение аргумента size для команд create database и alter database интерпретируется в мегабайтах дискового пространства. Это значение преобразуется в число логических страниц, которое было задано при создании главного устройства.
- Минимальный размер фрагмента дискового пространства, который можно инициализировать с помощью команды disk init, представляет собой наибольшее из следующих значений:
  - Один мегабайт
  - Одна единица выделения памяти с размером логической страницы сервера.

- Необходимо использовать команду disk init для каждого нового устройства хранения базы данных. При каждом вызове команды disk init в таблицу master..sysdevices добавляется одна строка. Новое устройство хранения базы данных не включается автоматически в пул устройств, используемых для хранения файлов базы данных по умолчанию. Для присвоения устройству хранения базы данных статуса “по умолчанию” используется процедура sp_diskdefault.
- После каждого вызова команды disk init необходимо создать резервную копию базы данных master с помощью команды dump database или dump transaction. Это упрощает восстановление базы данных master в случае ее повреждения и делает это восстановление более надежным. Если после добавления устройства с помощью команды disk init не было выполнено резервное копирование базы данных master, возможно, вам удастся восстановить изменения с помощью следующей процедуры: выполнения команды disk reinit, а затем остановки и повторного запуска сервера Adaptive Server.
- Для назначения устройств хранения базы данных для пользовательских баз данных используется инструкция name команды create database или alter database.
- Чтобы сохранить журнал транзакций базы данных (системная таблица syslogs) не на том устройстве, где размещается остальная часть базы данных, а на другом, лучше использовать дополнительную инструкцию log on для команды create database. В качестве альтернативного варианта при создании базы данных можно задать имена как минимум двух устройств и вызвать процедуру sp_logdevice. Кроме того, можно использовать команду alter database для расширения базы данных и задания для нее второго устройства, а затем вызвать процедуру sp_logdevice. Дополнительная инструкция log on сразу переносит весь журнал на отдельное устройство. Процедура sp_logdevice сохраняет часть системного журнала на исходном устройстве хранения базы данных до тех пор, пока информация о транзакциях не заполнит его настолько, что он целиком переместится на новое место.
- Чтобы получить отчет по всем устройствам СУБД Adaptive Server на вашей системе (по устройствам хранения как файлов баз данных, так и их резервных копий), нужно вызвать процедуру sp_helpdevice.
- Для удаления устройства хранения базы данных служит процедура sp_dropdevice. (Предварительно нужно удалить все базы данных на этом устройстве.)

После удаления устройства хранения базы данных можно создать новое с тем же именем (с помощью команды disk init), если задать для него другое физическое имя (размещение) и другой номер виртуального устройства. Чтобы использовать те же самые физическое имя и номер виртуального устройства, необходимо перезапустить сервер Adaptive Server.

- Если команда disk init дала сбой из-за того, что было указано слишком большое значение size для конкретного устройства хранения базы данных, то следует задать другой номер виртуального устройства или перезапустить сервер Adaptive Server перед повторным выполнением команды disk init.

Использование параметра *dsync*

Примечание. Нельзя присваивать параметру dsync значение false для устройств, на которых хранятся важные данные. Единственным исключением является база данных tempdb, которая без опасений может быть сохранена на устройствах, параметр dsync для которых имеет значение false.

- Если параметр dsync включен, при записи на устройство хранения базы данных информация всегда записывается прямо на носитель, и в случае системных сбоев сервер Adaptive Server сможет восстановить данные на этом устройстве.
- Если параметр dsync выключен, во время записи на устройство хранения базы данных файловая система UNIX может поместить данные в буфер. Файловая система UNIX может помечать обновление как завершенное даже тогда, когда данные на физическом носителе еще не были изменены. В случае системного сбоя нет никакой гарантии, что обновление данных произошло на физическом уровне, и их восстановление может завершиться неудачей.
- Параметр dsync всегда включен для файлов главного устройства.
- Параметр dsync можно отключать только в том случае, если базы данных, находящиеся на устройстве, можно не восстанавливать после системного сбоя. Например, можно отключить его для устройства, на котором хранится только база данных tempdb.
- Сервер Adaptive Server игнорирует значение параметра dsync для устройств, которые хранятся в неотформатированных (raw) разделах; при записи на эти устройства данные всегда записываются прямо на носитель, независимо от значения dsync.

- Параметр dsync не используется при работе с платформой Windows NT.
- С помощью команды disk reinit можно обеспечить достоверность данных в таблице master..sysdevices, если база данных master была повреждена или после последнего сохранения резервной копии базы данных master добавлялись новые устройства.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на запуск команды disk init принадлежат системным администраторам и не могут быть предоставлены другим пользователям. Команду disk init нужно выполнять из базы данных master.

См. также

Команды alter database, create database, disk refit, disk reinit, dump database, dump transaction, load database, load transaction

Системные процедуры sp_diskdefault, sp_dropdevice, sp_helpdevice, sp_logdevice

disk mirror

Описание

Создает зеркальную копию файлов баз данных, которая сразу включается в работу в случае сбоя основного устройства.

Синтаксис

disk mirror
    name = "*имя_устройства*" ,
    mirror = "*размещение_устройства*"
    [, writes = { serial | noserial }]

Параметры

name

Имя устройства хранения базы данных, зеркальную копию которого нужно создать. Оно записывается в столбец name таблицы sysdevices. Это имя должно быть заключено в одинарные или двойные кавычки.

mirror

Полный путь к устройству – зеркальной копии, которое будет вторичным устройством. Его нужно заключать в одинарные или двойные кавычки. Если вторичное устройство представляет собой файл, параметр physicalname должен задавать полный путь к файлу, создаваемому сервером Adaptive Server. В этом параметре нельзя указывать уже существующий файл.

writes

Указывает, должна ли запись на устройства выполняться в последовательном режиме. Если используется значение по умолчанию (serial, последовательная запись), запись на вторичное устройство хранения базы данных всегда будет выполняться только после того, как завершится запись на первичное устройство. Если первичное и вторичное устройства находятся на разных физических устройствах, последовательная запись может гарантировать, что в случае сбоя подачи питания удастся сохранить данные по крайней мере на одном диске.

Примеры

```
disk mirror
    name = "user_disk",
    mirror = "/server/data/mirror.dat"
```

Указание, что данные устройства user_disk будут зеркально копироваться в файл *mirror.dat*.

Использование

- Зеркальное копирование диска означает, что создается копия информации, содержащейся на устройствах хранения пользовательских баз данных, главном устройстве хранения базы данных или устройстве, на котором хранятся журналы транзакций пользовательских баз данных. В случае сбоя устройства хранения базы данных в работу сразу же включается его зеркальная копия.

Зеркальное копирование диска не мешает текущим операциям, выполняющимся в базе данных. Можно запускать и останавливать зеркальное копирование для устройств хранения базы данных, не прекращая работу Adaptive Server.

- После каждого вызова команды disk mirror необходимо делать резервное копирование базы данных master с помощью команды dump database. Это упрощает восстановление базы данных master в случае ее повреждения и повышает вероятность благополучного исхода такого восстановления.
- При неудачном чтении или записи на устройство, для которого существует зеркальная копия, сервер Adaptive Server останавливает зеркальное копирование для этого устройства и выводит сообщения об ошибках. После этого работа продолжается в режиме без зеркального копирования. Чтобы повторно запустить зеркальное копирование, системный администратор должен выполнить команду disk remirror.
- Зеркальное копирование можно использовать для главного устройства и для устройств, где хранятся данные и журналы транзакций. Но для устройств резервного копирования его использовать нельзя.
- Зеркальные копии можно создавать для устройств, но не для баз данных.
- Устройство и его зеркальная копия образуют одно логическое устройство. СУБД Adaptive Server хранит физическое имя зеркального устройства в столбце mirrorname таблицы sysdevices. Оно не требует отдельной записи в таблице sysdevices и не должно инициализироваться командой disk init.
- Чтобы иметь возможность использовать асинхронный ввод-вывод, следует организовывать зеркальное копирование так, чтобы и первичное устройство, и его зеркальная копия поддерживали такой механизм ввода-вывода. В большинстве случаев это означает, что в качестве зеркальных копий для неотформатированных устройств нужно использовать неотформатированные устройства, а для файлов операционной системы – файлы операционной системы.

  Если операционная система не поддерживает асинхронный ввод-вывод в файлы, зеркальное копирование неотформатированного устройства в обычный файл приводит к появлению сообщения об ошибке. Если имеет место обратная ситуация, сообщения об ошибке не возникает, но в этом случае не используется асинхронный ввод-вывод.

- Создание зеркальных копий для всех устройств хранения базы данных, используемых по умолчанию, позволяет восстановить данные после ошибочного применения команды create database или alter database для одного из таких устройств.
- Чтобы обеспечить большую надежность в хранении данных, рекомендуется создавать зеркальные копии для устройств, используемых для хранения журналов транзакций.
- Журналы транзакций пользовательских баз данных должны всегда размещаться на отдельном устройстве хранения базы данных. Чтобы поместить журнал транзакций базы данных (то есть системную таблицу syslogs) не на то устройство, где хранится остальная часть базы данных, а на другое, при создании базы данных укажите имена устройств хранения базы данных и ее журнала. Можно также использовать команду alter database для расширения базы данных и задания для нее второго устройства, а затем вызвать процедуру sp_logdevice.
- Если создается зеркальная копия устройства, хранящего базу данных master, при перезапуске сервера Adaptive Server в утилите dataserver можно указать параметр -r и имя зеркальной копии в UNIX. Этот параметр нужно добавить в файл *RUN_servername* данного сервера, чтобы утилита startserver смогла прочесть его. Например, чтобы запустить главное устройство с именем master.dat и его зеркальную копию mirror.dat, введите следующую команду:

  ```
  dataserver -dmaster.dat -rmirror.dat
  ```

  Дополнительную информацию см. в описании утилит dataserver и startserver в книге *Utility Guide*.
- Если выполняется зеркальное копирование устройства баз данных, на котором есть нераспределенные области (то есть пространство, которое могут использовать команды create database и alter database), выделение пространства в этих областях отражается в зеркальной копии тогда, когда оно фактически производится, а не при запуске команды disk mirror.
- Для получения отчета обо всех устройствах СУБД Adaptive Server, имеющихся в системе (устройствах хранения пользовательских баз данных и их зеркальных копиях, а также об устройствах резервного копирования), выполните процедуру sp_helpdevice.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на запуск команды disk mirror принадлежат системным администраторам и не могут быть предоставлены другим пользователям. Чтобы запустить команду disk mirror, вы должны использовать базу данных master.

См. также

Команды alter database, create database, disk init, disk refit, disk reinit, disk remirror, disk unmirror, dump database, dump transaction, load database, load transaction

Системные процедуры sp_diskdefault, sp_helpdevice, sp_logdevice

Утилиты dataserver, startserver

disk refit

Описание

Перестраивает системные таблицы sysusages и sysdatabases базы данных master на основании информации, содержащейся в таблице sysdevices.

Синтаксис

disk refit

Примеры

```
disk refit
```

Использование

- Сервер Adaptive Server автоматически прекращает работу после того, как команда disk refit перестраивает системные таблицы.
- Команда disk refit используется после команды disk reinit как часть процесса восстановления базы данных master. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

Примечание. Чтобы выполнять команду disk refit, нужно запустить сервер Adaptive Server с флагом трассировки 3608. Однако перед тем как использовать флаги трассировки при запуске сервера Adaptive Server, необходимо ознакомится с материалами книги *Troubleshooting and Error Messages Guide*.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на запуск команды disk refit принадлежат системным администраторам и не могут быть предоставлены другим пользователям. Чтобы использовать команду disk refit, нужно, чтобы текущей была база данных master.

См. также

Команды disk init, disk reinit

Системные процедуры sp_addumpdevice, sp_helpdevice

disk reinit

Описание

Перестраивает системную таблицу sysdevices базы данных master. Команда disk reinit используется в процессе восстановления базы данных master.

Синтаксис

```
disk reinit
        name = "имя_устройства" ,
        physname = "размещение_устройства" ,
        [vdevno = номер_виртуального_устройства ,]
        size = число_блоков
        [, vstart = виртуальный_адрес
            , cntrltype = номер_контроллера ]
        [, dsync = { true | false } ]
```

Параметры

name

Имя устройства хранения базы данных. Должно соответствовать правилам именования для идентификаторов и должно быть заключено в одинарные или двойные кавычки. Это имя используется в командах create database и alter database.

physname

Имя устройства хранения базы данных. Физическое имя должно быть заключено в одинарные или двойные кавычки.

vdevno

Номер виртуального устройства. Должен быть уникальным для любого устройства, используемого сервером Adaptive Server. Номер устройства 0 зарезервирован для устройства хранения базы данных master. Можно использовать номера в диапазоне от 1 до 255, но номер не может быть больше общего количества устройств хранения базы данных, на которое настроена система. Значение по умолчанию – 50 устройств.

size

Размер области, которая выделяется при расширении базы данных. Размер может быть задан в следующих единицах: ‘k’ или ‘K’ (килобайты), ‘m’ или ‘M’ (мегабайты) и ‘g’ или ‘G’ (гигабайты). Рекомендуется всегда указывать единицы измерения. Если единица измерения не указана, кавычки можно опустить. Но если она указана, кавычки нужны.

vstart

Начальный виртуальный адрес (смещение), начиная с которого сервер Adaptive Server будет использовать пространство устройства хранения базы данных. В параметре vstart можно указывать следующие обозна-

чения единиц измерения пространства: ‘k’ или ‘K’ (килобайты), ‘m’ или ‘M’ (мегабайты) и ‘g’ или ‘G’ (гигабайты). Величина смещения зависит от того, каким образом введено значение параметра vstart.

- Если единица измерения не указана, для вычисления начального адреса используются страницы размером 2 КБ. Например, если задано `vstart = 13`, то сервер Adaptive Server использует в качестве смещения для начального адреса величину, равную 13 * 2 КБ.
- Если единица измерения указана, то в качестве начального адреса используется заданное значение. Например, если задано `vstart = "13M"`, сервер Adaptive Server устанавливает для начального адреса смещение, равное 13 мегабайтам.

Значение по умолчанию (и, как правило, предпочтительное значение) для параметра vstart равно 0. Если на заданном устройстве количество доступных блоков меньше суммы значений vstart + size, выполнение команды disk reinit заканчивается неудачей.

Примечание. Если вы работаете в операционной системе AIX и используете утилиту Logical Volume Manager, значение параметра vstart должно равняться 2.

Задавать параметру vstart другие значения можно только в том случае, если такое указание было дано службой технической поддержки Sybase.

cntrltype

Контроллер диска. Значение по умолчанию равно 0. Изменять значение этого параметра следует только в том случае, если такое указание было дано службой технической поддержки Sybase.

dsync

Только для платформ UNIX. Если используются файлы операционной системы UNIX, этот параметр определяет, будут ли операции записи на устройство хранения базы данных записывать информацию прямо на носитель или использовать буферизацию. Использовать этот параметр имеет смысл только в том случае, если инициализируемый файл находится в операционной системе UNIX. Он не оказывает никакого действия при инициализации устройств в неотформатированных (raw) разделах. По умолчанию при инициализации всех файлов операционной системы UNIX параметр dsync имеет значение true.

Примеры

Инициализация 10 МБ дискового пространства в файле операционной системы UNIX. СУБД Adaptive Server открывает файл, используя заданное значение параметра dsync, и при записи в файл данные гарантированно записываются прямо на носитель:

```
disk reinit
name = "user_file",
physname = "/usr/u/sybase/data/userfile1.dat",
vdevno = 2, size = 5120, dsync = true
```

Использование

- С помощью команды disk reinit можно обеспечить достоверность данных в таблице master..sysdevices, если база данных master была повреждена или после последнего сохранения ее резервной копии добавлялись новые устройства.
- Команда disk reinit похожа на disk init, но она не инициализирует устройство хранения базы данных.
- Размер можно задавать в виде числа с плавающей точкой, но оно будет округлено в сторону уменьшения до ближайшего числа, кратного 2 КБ.
- Если единица измерения для параметра size не указана, команда disk reinit использует виртуальные страницы размером 2 КБ.
- Полная информация о восстановлении базы данных master содержится в книге *Руководство по системному администрированию*.

Использование параметра *dsync*

Примечание. Нельзя присваивать параметру dsync значение false для устройств, на которых хранятся важные данные. Единственным исключением является база данных tempdb, которая без опасений может быть сохранена на устройствах, параметр dsync для которых имеет значение false.

- Если параметр dsync включен, при записи на устройство хранения базы данных информация всегда записывается прямо на носитель, и в случае системных сбоев сервер Adaptive Server сможет восстановить данные на этом устройстве.
- Если параметр dsync выключен, во время записи на устройство хранения базы данных файловая система UNIX может поместить данные в буфер. Файловая система UNIX может помечать обновление как завершенное даже тогда, когда данные на физическом носителе еще не были изменены. В случае системного сбоя нет никакой гарантии, что обновление данных произошло на физическом уровне, и их восстановление может завершиться неудачей.

- Параметр dsync всегда включен для файлов главного устройства.
- Параметр dsync можно отключать только в том случае, если базы данных, находящиеся на устройстве, можно не восстанавливать после системного сбоя. Например, можно отключить его для устройства, на котором хранится только база данных tempdb.
- Сервер Adaptive Server игнорирует значение параметра dsync для устройств, которые хранятся в неотформатированных (raw) разделах; при записи на эти устройства данные всегда записываются прямо на носитель, независимо от значения dsync.
- Параметр dsync не используется при работе с платформой Windows NT.
- С помощью команды disk reinit можно обеспечить достоверность данных в таблице master..sysdevices, если база данных master была повреждена или после последнего сохранения ее резервной копии добавлялись новые устройства.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на запуск команды disk reinit принадлежат системным администраторам и не могут быть предоставлены другим пользователям. Чтобы использовать команду disk reinit, нужно, чтобы текущей была база данных master.

См. также

Команды alter database, create database, dbcc, disk init, disk refit

Системные процедуры sp_addumpdevice, sp_helpdevice

disk remirror

Описание

Повторно запускает зеркальное копирование диска, после того как оно было прекращено из-за сбоя устройства, копия которого создается, или временно остановлено командой disk unmirror.

Синтаксис

disk remirror
name = "*имя_устройства*"

Параметры

name

Имя устройства хранения базы данных, для которого нужно возобновить ведение резервной копии. Это имя записывается в столбец name таблицы sysdevices и должно быть заключено в одинарные или двойные кавычки.

Примеры

Возобновление зеркального копирования информации с устройства user_disk:

```
disk remirror
  name = "user_disk"
```

Использование

- При зеркальном копировании диска создается зеркальная копия устройств хранения пользовательских баз данных, главного устройства хранения базы данных или устройств, на которых хранятся журналы транзакций пользовательских баз данных. В случае сбоя устройства хранения базы данных в работу сразу же включается его зеркальная копия.

  Команда disk remirror используется для возобновления зеркального копирования после того, как оно было приостановлено из-за сбоя устройства, для которого велась зеркальная копия, или в результате выполнения команды disk unmirror с параметром mode = retain. Команда disk remirror возобновляет копирование данных на зеркальное устройство.

- После каждого вызова команды disk remirror необходимо сохранить резервную копию базы данных master с помощью команды dump database. Это упрощает восстановление базы данных master в случае ее повреждения и повышает вероятность благополучного исхода такого восстановления.

- Если зеркальное копирование было полностью отключено с помощью параметра mode = remove, перед запуском команды disk remirror нужно удалить файл операционной системы, который содержит зеркальную копию.

- Зеркальное копирование применяется к устройствам хранения баз данных, а не к самим базам данных.

- Запускать зеркальное копирование для устройств хранения базы данных, возобновлять и останавливать его можно, не прекращая работу сервера Adaptive Server. Зеркальное копирование диска не мешает текущим операциям, выполняющимся в базе данных.
- При неудачном чтении или записи на устройство, для которого существует зеркальная копия, сервер Adaptive Server останавливает зеркальное копирование для этого устройства и выводит сообщения об ошибках. После этого работа продолжается в режиме без зеркального копирования. Чтобы повторно запустить зеркальное копирование, системный администратор должен выполнить команду disk remirror.
- Помимо зеркального копирования устройств хранения пользовательских баз данных, необходимо разместить журналы транзакций таких баз данных на отдельном устройстве. Для устройства хранения баз данных, используемого для журналов транзакций, также может быть создана зеркальная копия, что еще более повышает степень защиты данных. Чтобы поместить журнал транзакций базы данных (то есть системную таблицу syslogs) не на то устройство, где хранится остальная часть базы данных, а на другое, при создании базы данных укажите имена устройств хранения базы данных и ее журнала. Можно также запустить команду alter database для второго устройства и выполнить процедуру sp_logdevice.
- Если создается зеркальная копия устройства хранения БД для базы данных master, в системах UNIX имя устройства – зеркальной копии можно задать, используя параметр -r при перезапуске сервера Adaptive Server с помощью утилиты dataserver. Добавьте этот параметр в файл *RUN_servername*, предназначенный для данного сервера, чтобы утилита startserver смогла прочесть его. Например, следующая команда запускает главное устройство с именем master.dat и его зеркальную копию mirror.dat:

  ```
  dataserver -dmaster.dat -rmirror.dat
  ```

  Дополнительную информацию см. в описании утилит dataserver и startserver в книге *Utility Guide*.
- Для получения отчета обо всех устройствах СУБД Adaptive Server, имеющихся в системе (устройствах хранения пользовательских баз данных и их зеркальных копиях, а также об устройствах резервного копирования), запустите процедуру sp_helpdevice.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на запуск команды disk remirror принадлежат системным администраторам и не могут быть предоставлены другим пользователям. Чтобы запустить команду disk remirror, текущей базой данных должна быть база данных master.

См. также

Команды alter database, create database, disk init, disk mirror, disk refit, disk reinit, disk unmirror, dump database, dump transaction, load database, load transaction

Системные процедуры sp_diskdefault, sp_helpdevice, sp_logdevice

Утилиты dataserver, startserver

disk unmirror

Описание

Приостанавливает зеркальное копирование диска, запущенное командой disk mirror, что позволяет произвести обслуживание или замену оборудования.

Синтаксис

```
disk unmirror
        name = "имя_устройства"
        [ ,side = { "primary" | secondary }]
        [ ,mode = { retain | remove }]
```

Параметры

name

Имя устройства хранения базы данных, для которого нужно приостановить зеркальное копирование. Это имя должно быть заключено в одинарные или двойные кавычки.

side

Определяет, будет ли отключено первичное (primary) или вторичное (secondary) устройство (зеркальная копия). По умолчанию зеркальное копирование приостанавливается для вторичного устройства.

mode

Определяет, будет ли приостановка зеркального копирования временной (retain) или постоянной (remove). По умолчанию приостановка является временной.

Если планируется в будущем возобновить зеркальное копирование для этого устройства хранения базы данных в той же конфигурации, то необходимо указать параметр retain. Этот параметр имитирует ситуацию, когда происходит сбой первичного устройства:

- Ввод-вывод осуществляется только на устройство, для которого *не* отключено зеркальное копирование.
- Столбец status в таблице sysdevices указывает, что зеркальное копирование отключено. Если используется параметр remove, из таблицы sysdevices удаляются все ссылки на зеркальное устройство.
- Столбец status показывает, что возможность зеркального копирования игнорируется.
- В столбец phyname записывается имя вторичного устройства из столбца mirrorname, если зеркальное копирование приостанавливается для первичного устройства.
- Столбцу mirrorname присваивается значение NULL.

Примеры

Пример 1. Приостановка зеркального копирования данных для устройства хранения базы данных user_disk:

```
disk unmirror
name = "user_disk"
```

Пример 2. Приостановка зеркального копирования данных для вторичного устройства хранения базы данных user_disk:

```
disk unmirror name = "user_disk", side = secondary
```

Пример 3. Приостановка зеркального копирования данных для устройства хранения базы данных user_disk и удаление всех ссылок на зеркальное устройство из таблицы sysdevices:

```
disk unmirror name = "user_disk", mode = remove
```

Использование

- Зеркальное копирование диска означает, что создается копия информации, содержащейся на устройствах хранения пользовательских баз данных, главном устройстве хранения базы данных или устройстве, на котором хранятся журналы транзакций пользовательских баз данных. В случае сбоя устройства хранения базы данных в работу сразу же включается его зеркальная копия.

  Команда disk unmirror временно или постоянно отключает либо само устройство хранения базы данных, либо его зеркальную копию, и сервер Adaptive Server не может считывать и записывать данные на это устройство. Она не удаляет из операционной системы файл, связанный с этим устройством.

- При отключении зеркального копирования диска вносятся изменения в таблицу sysdevices в базе данных master. После каждого вызова команды disk unmirror необходимо создавать резервную копию базы данных master с помощью команды dump database. Это упрощает восстановление базы данных master в случае ее повреждения и делает это восстановление более надежным.

- Можно останавливать зеркальное копирование устройств хранения баз данных во время их использования.

- Нельзя отключить зеркальное копирование какого-либо устройства хранения базы данных в то время, когда выполняется команда dump database, load database или load transaction. Если попытаться сделать это, будет выведено сообщение с вопросом, какие действия следует предпринять: остановить создание или загрузку резервной копии или отложить выполнение команды disk unmirror до тех пор, пока не завершится создание или загрузка резервной копии.

- Нельзя отключить зеркальное копирование устройства, на котором хранится журнал базы данных, во время выполнения команды dump

transaction. Если попытаться сделать это, будет выведено сообщение с вопросом, следует ли остановить создание резервной копии или отложить выполнение команды disk unmirror до тех пор, пока создание резервной копии не завершится.

Примечание. Отключение зеркального копирования устройства для хранения журнала не влияет на выполнение команд dump transaction with truncate_only и dump transaction with no_log.

- Следует создавать зеркальные копии для всех устройств хранения базы данных, используемых по умолчанию, чтобы была возможность восстановить данные после того, как база данных была ошибочно помещена на одно из таких устройств командой create или alter database.
- При неудачном чтении или записи на устройство, для которого существует зеркальная копия, сервер Adaptive Server останавливает зеркальное копирование для этого устройства и выводит сообщения об ошибках. После этого работа продолжается в режиме без зеркального копирования. Системный администратор должен повторно запустить зеркальное копирование с помощью команды disk remirror.
- Для получения отчета обо всех устройствах СУБД Adaptive Server, имеющихся в системе (устройствах хранения пользовательских баз данных и их зеркальных копиях, а также об устройствах для хранения резервных копий), выполните процедуру sp_helpdevice.
- Команда disk remirror используется для возобновления зеркального копирования после его приостановки с помощью параметра mode = retain команды disk unmirror. Если зеркальное копирование отключено “навсегда” (то есть указан параметр mode = remove), то перед запуском команды disk remirror нужно удалить файл операционной системы, который содержит зеркальную копию.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на запуск команды disk unmirror принадлежат системному администратору и не могут быть предоставлены другим пользователям. Команду disk unmirror можно выполнять только из базы данных master.

См. также

Команды alter database, create database, disk init, disk mirror, disk refit, disk reinit, disk remirror, dump database, dump transaction, load database, load transaction

Системные процедуры sp_diskdefault, sp_helpdevice, sp_logdevice

Утилиты dataserver, startserver

drop database

Описание

Удаляет одну или несколько баз данных с сервера Adaptive Server.

Синтаксис

drop database *имя_базы_данных* [, *имя_базы_данных*] ...

Параметры

имя_базы_данных

Имя удаляемой базы данных. Для отображения списка баз данных используется системная процедура sp_helpdb.

Примеры

Удаление базы данных publishing и всего ее содержимого:

```
drop database publishing
```

Использование

- При удалении базы данных удаляются сама база данных и все ее объекты, освобождается выделенное для ее хранения пространство и удаляются относящиеся к ней записи из системных таблиц sysdatabases и sysusages в базе данных master.
- Команда drop database очищает записи подозрительных страниц, относящиеся к удаленной базе данных, из таблицы master..sysattributes.

Ограничения

- Команду drop database можно выполнять только из базы данных master.
- Нельзя удалять используемую базу данных (открытую каким-либо пользователем для чтения или записи).
- Нельзя использовать команду drop database для удаления базы данных, на которую ссылается таблица в другой базе данных. Чтобы определить, в каких таблицах и внешних базах данных определены ограничения внешнего ключа, ссылающиеся на первичные ключи таблиц текущей базы данных, выполните следующий запрос:

  ```
  select object_name(tableid), frgndbname
      from sysreferences
      where frgndbname is not null
  ```

  Эти межбазовые ограничения можно удалить командой alter table, после чего снова выполнить команду drop database.
- Команду drop database нельзя использовать для удаления поврежденной базы данных. Для этого нужно сначала выполнить команду dbcc dbrepair:

  ```
  dbcc dbrepair (имя_базы_данных, dropdb)
  ```

- Нельзя удалять базу данных sybsecurity, если включен аудит. Если аудит отключен, удалить базу данных sybsecurity может только администратор безопасности системы.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду drop database может выполнять только владелец базы данных; исключение составляет база данных sybsecurity, удалить которую может только администратор безопасности системы.

См. также

Команды alter database, create database, dbcc, use

Утилиты sp_changedbowner, sp_helpdb, sp_renamedb, sp_spaceused

drop default

Описание

Удаляет определенные пользователем значения по умолчанию.

Синтаксис

drop default [*владелец*.]*имя_по_умолчанию*
[, [*владелец*.]*имя_по_умолчанию*] ...

Параметры

имя_по_умолчанию

Имя существующего значения по умолчанию. Для отображения списка значений по умолчанию используется системная процедура sp_helpdb. Чтобы удалить значение по умолчанию, принадлежащее другому пользователю в текущей базе данных, перед именем значения по умолчанию нужно указать имя владельца. По умолчанию *владелец* – текущий пользователь.

Примеры

Удаление определенного пользователем значения по умолчанию datedefault из базы данных:

```
drop default datedefault
```

Использование

- Нельзя удалить значение по умолчанию, которое в текущий момент назначено столбцу или типу данных, определенному пользователем. В этом случае перед удалением этого значения по умолчанию нужно отменить эту привязку с помощью системной процедуры sp_unbindefault.
- Столбцу или пользовательскому типу данных можно назначить новое значение по умолчанию, не отменяя текущего значения по умолчанию. В этом случае новое значение по умолчанию заменит старое.
- Если значение по умолчанию удаляется для столбца, для которого допустимы значения NULL, то новым значением по умолчанию для этого столбца становится NULL. Если же значение по умолчанию удаляется для столбца NOT NULL, то будет выдана ошибка, если пользователь не введет явным образом значение этого столбца при вставке данных.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на выполнение команды drop default принадлежат владельцу этого значения по умолчанию и их нельзя передать другим пользователям.

См. также

Команды create default

Системные процедуры sp_help, sp_helptext, sp_unbindefault

drop function (SQLJ)

Описание

Эта команда удаляет функцию SQLJ.

Синтаксис

drop func[tion] [*владелец*.]*имя_функции*
[, [*владелец*.]*имя_функции*] ...

Параметры

[владелец.]имя_функции
Имя SQL функции SQLJ.

Примеры

Удаление функции SQLJ square_root:

```
drop function square_root
```

Использование

- Дополнительную информацию о функциях SQLJ см. в книге *Java in Adaptive Server Enterprise*.
- Команда drop function может удалять только созданные пользователем функции и только из текущей базы данных. Она не удаляет системные функции.

Полномочия

Команду drop function может выполнять только владелец базы данных или пользователь с ролью sa.

См. также

Команды create function (SQLJ)

drop index

Описание

Удаляет индекс из таблицы в текущей базе данных.

Синтаксис

drop index *имя_таблицы.имя_индекса*
[, *имя_таблицы.имя_индекса*] ...

Параметры

имя_таблицы
Имя таблицы, содержащей индексированный столбец. Эта таблица должна находиться в текущей базе данных.

имя_индекса
Имя удаляемого индекса. В Transact-SQL имена индексов не обязательно должны быть уникальными во всей базе данных, но должны быть уникальными внутри одной таблицы.

Примеры

Удаление индекса au_id_ind из таблицы authors:

```
drop index authors.au_id_ind
```

Использование

- Команда drop index освобождает все пространство, которое занимал до этого индекс. Это пространство может использоваться для любых объектов базы данных.
- Нельзя использовать команду drop index для системных таблиц.
- Команда drop index не может удалять индексы, которые обеспечивают соблюдение ограничения unique. Чтобы удалить такие индексы, нужно сначала удалить ограничения командой alter table или удалить таблицу. Дополнительную информацию об индексах, обеспечивающих соблюдение ограничения unique, см. в описании команды create table.
- Нельзя удалять индексы, которые в настоящее время используются каким-либо открытым курсором. Информацию о том, какие курсоры открыты и какие индексы они используют, можно получить с помощью системной процедуры sp_cursorinfo.
- Чтобы узнать о том, какие индексы были определены для таблицы, выполните следующую системную процедуру, где имя_объекта – имя таблицы:

```
sp_helpindex имя_объекта
```

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Полномочия на удаление индекса с помощью команды drop index по умолчанию принадлежат владельцу индекса и не могут передаваться.

См. также

Команды create index

Системные процедуры sp_cursorinfo, sp_helpindex, sp_spaceused

drop procedure

Описание

Удаляет процедуру.

Синтаксис

drop proc[edure] [*владелец*.]*имя_процедуры*
[, [*владелец*.]*имя_процедуры*] ...

Параметры

имя_процедуры

Имя удаляемой процедуры Transact-SQL или SQLJ. Чтобы удалить процедуру, принадлежащую другому пользователю в текущей базе данных, перед именем процедуры нужно указать имя владельца. По умолчанию *владелец* – это текущий пользователь.

Примеры

Пример 1. Удаление хранимой процедуры showind:

```
drop procedure showind
```

Пример 2. Удаление расширенной хранимой процедуры xp_echo:

```
drop procedure xp_echo
```

Использование

- Команда drop procedure удаляет определенные пользователем хранимые процедуры, системные процедуры и расширенные хранимые процедуры (extended stored procedure, ESP).
- Каждый раз, когда пользователь или программа пытаются выполнить процедуру, Adaptive Server проверяет, существует ли эта процедура.
- С помощью одной команды drop procedure можно удалить группу процедур (несколько процедур с одинаковыми именами, но разными числовыми суффиксами). Например, если процедуры, используемые в приложении orders, имеют имена orderproc;1, orderproc;2 и т.д, следующая команда удалит всю группу:

  ```
  drop proc orderproc
  ```

  После того как процедуры были сгруппированы, отдельную процедуру, принадлежащую группе процедур, нельзя удалить. Например, следующая команда недопустима:

  ```
  drop procedure orderproc;2
  ```

  Нельзя удалять расширенные хранимые процедуры как группу процедур.
- Текст процедуры, хранящийся в таблице syscomments, можно получить с помощью системной процедуры sp_helptext.
- Имена расширенных хранимых процедур и соответствующие им библиотеки DLL можно вывести с помощью системной процедуры sp_helpextendedproc.

- Удаление расширенной хранимой процедуры отменяет ее регистрацию, удаляя ее из системных таблиц, но не влияет на библиотеку DLL, на которой основана эта процедура.
- Команда drop procedure может удалять только созданные пользователем процедуры и только из текущей базы данных.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Полномочия на выполнение команды drop procedure по умолчанию принадлежат владельцу процедуры и не могут передаваться.

См. также

Команды create procedure, create procedure (SQLJ)

Системные процедуры sp_depends, sp_dropextendedproc, sp_helpextendedproc, sp_helptext, sp_rename

drop role

Описание

Удаляет определенную пользователем роль.

Синтаксис

drop role *имя_роли* [with override]

Параметры

имя_роли

Имя роли, которую нужно удалить.

with override

Переопределяет любые ограничения на удаление роли. Если указан параметр with override, роль будет удалена без проверки того, были ли удалены предоставленные этой ролью полномочия во всех базах данных.

Примеры

Пример 1. Эта команда удаляет указанную роль, только если были отозваны все полномочия во всех базах данных. Поэтому прежде чем удалять роль, системный администратор или владелец объекта должны отозвать предоставленные полномочия во всех базах данных (в противном случае команда не будет успешно выполнена):

```
drop role doctor_role
```

Пример 2. Удаление из всех баз данных указанной роли, информации о полномочиях и всех остальных ссылок на эту роль:

```
drop role doctor_role with override
```

Использование

- Перед удалением роли не нужно исключать пользователей из этой роли. Команда drop role делает это автоматически (независимо от того, был ли указан параметр with override).
- Команда drop role выполняется в базе данных master.

Ограничения

- Нельзя использовать команду drop role для удаления системных ролей.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Выполнять команду drop role может только администратор безопасности системы.

Полномочия на выполнение команды drop role не предоставляются командой grant all.

См. также

Команды alter role, create role, grant, revoke, set

Системные процедуры sp_activeroles, sp_displaylogin, sp_displayroles, sp_helprotect, sp_modifylogin

drop rule

Описание

Удаляет правило, определенное пользователем.

Синтаксис

drop rule [*владелец*.]*имя_правила* [, [*владелец*.]*имя_правила*] ...

Параметры

имя_правила

Имя правила, которое нужно удалить. Чтобы удалить правило с таким же именем, принадлежащее другому пользователю в текущей базе данных, перед именем правила нужно указать имя владельца. По умолчанию *владелец* – текущий пользователь.

Примеры

Удаление из текущей базы данных правила pubid_rule:

```
drop rule pubid_rule
```

Использование

- Перед удалением правила необходимо отменить его привязку с помощью системной процедуры sp_unbindrule. Если связывание правила не отменено, выводится сообщение об ошибке и команда drop rule завершается со сбоем.
- Можно связать новое правило со столбцом или определенным пользователем типом данных, не отменяя связывания текущего правила. Новое правило переопределяет старое.
- После удаления правила сервер Adaptive Server вводит новые данные в столбцы, которые до этого управлялись этим правилом, без ограничений. Существующие данные не изменяются во всех случаях.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Полномочия на удаление правила с помощью команды drop rule по умолчанию принадлежат владельцу правила и не могут передаваться.

См. также

Команды create rule

Системные процедуры sp_bindrule, sp_help, sp_helptext, sp_unbindrule

drop table

Описание

Удаляет из базы данных определение таблицы и все относящиеся к ней данные, индексы, триггеры и полномочия.

Синтаксис

drop table [[*база_данных*.]*владелец*.]*имя_таблицы*
[, [[*база_данных*.]*владелец*.]*имя_таблицы*] ...

Параметры

имя_таблицы

Имя таблицы, которую необходимо удалить. Если таблица находится в другой базе данных, то необходимо указать имя этой базы данных, а если в базе данных существуют несколько таблиц с таким именем – имя владельца. По умолчанию *владелец* – текущий пользователь, а *база_данных* – текущая база данных.

Примеры

Удаление из текущей базы данных таблицы roysched и всех относящихся к ней данных и индексов:

```
drop table roysched
```

Использование

- Команда drop table отменяет привязку правил и значений по умолчанию к этой таблице, а также автоматически удаляет все триггеры, связанные с таблицей. При повторном создании таблицы необходимо связать соответствующие правила и значения по умолчанию и повторно создать все триггеры.
- При удалении таблицы изменяются системные таблицы sysobjects, syscolumns, sysindexes, sysprotects и syscomments.
- Если включены службы интеграции компонентов Component Integration Services и если удаляемая таблица была создана командой create existing table, то таблица не удаляется с удаленного сервера. Вместо этого сервер Adaptive Server удаляет из системных таблиц ссылки на эту таблицу.

Ограничения

- Нельзя использовать команду drop table для системных таблиц.
- Владелец таблицы может удалять эту таблицу вне зависимости от того, в какой базе данных она находится. Например, все следующие команды удаляют таблицу newtable в базе данных otherdb:

```
drop table otherdb..newtable
drop table otherdb.yourname.newtable
```

- Если над таблицей была выполнена команда delete, удаляющая все строки в таблице, или команда truncate table, таблица останется, пока над ней не будет выполнена команда drop.

Удаление таблиц с межбазовыми ограничениями ссылочной целостности

- Когда создается межбазовое ограничение, в системной таблице sysreferences каждой базы данных сохраняется следующая информация:

Таблица 7-21. Информация об ограничениях ссылочной целостности, хранящаяся в системных таблицах

Информация, хранящаяся в системной таблице sysreferences	Столбцы с информацией о родительской таблице	Столбцы с информацией о дочерней таблице
Идентификаторы ключевых столбцов	с refkey1 по refkey16	с fokey1 по fokey16
Идентификатор таблицы	reftabid	tableid
Имя базы данных	pmrydbname	frgndbname

- Так как дочерняя таблица зависит от информации из родительской таблицы, то сервер Adaptive Server не разрешает:
  - удалять родительскую таблицу;
  - удалять содержащую ее внешнюю базу данных;
  - переименовывать любую из этих баз данных с помощью системной процедуры sp_renamedb.

  Системная процедура sp_helpconstraint позволяет определить, какие таблицы ссылаются на удаляемую таблицу. Для удаления ограничений перед повторным выполнением команды drop table используется команда alter table.
- При удалении дочерней таблицы или ее базы данных не возникнет проблем. Сервер Adaptive Server автоматически удаляет информацию о внешних ключах из родительской базы данных.
- При каждом добавлении или удалении межбазового ограничения целостности или удалении таблицы, содержащей межбазовое ограничение целостности, необходимо создать резервную копию *обеих* баз данных, участвующих в ограничении.

Предупреждение. Загрузка более ранних резервных копий этих баз данных может привести к повреждениям базы данных. Дополнительную информацию о загрузке баз данных с межбазовыми ограничениями ссылочной целостности см. в книге *Руководство по системному администрированию*.

Стандарты	Уровень сответствия стандарту SQL92: расширение Transact-SQL.
Полномочия	Полномочия на удаление таблицы командой drop table по умолчанию принадлежат владельцу таблицы и не могут передаваться.
См. также	**Команды** alter table, create table, delete, truncate table
	Системные процедуры sp_depends, sp_help, sp_spaceused

drop trigger

Описание

Удаляет триггер.

Синтаксис

drop trigger [*владелец*.]*имя_триггера*
[, [*владелец*.]*имя_триггера*] ...

Параметры

имя_триггера

Имя триггера, который нужно удалить. Чтобы удалить триггер с таким же именем, принадлежащий другому пользователю в текущей базе данных, перед именем триггера нужно указать имя владельца. По умолчанию *владелец* – текущий пользователь.

Примеры

Удаление из текущей базы данных триггера trigger1:

```
drop trigger trigger1
```

Использование

- Команда drop trigger удаляет триггер в текущей базе данных.
- Не нужно явно удалять триггер, чтобы создать новый триггер для той же операции (insert, update или delete). Новый триггер для той же операции переписывает предыдущий триггер, определенный для той же таблицы или столбца.
- Когда удаляется таблица, сервер Adaptive Server автоматически удаляет все связанные с ней триггеры.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Полномочия на удаление триггера с помощью команды drop trigger по умолчанию принадлежат владельцу триггера и не могут передаваться.

См. также

Команды create trigger

Системные процедуры sp_depends, sp_help, sp_helptext

drop view

Описание

Удаляет одно или несколько представлений из текущей базы данных.

Синтаксис

drop view [*владелец*.]*имя_представления* [, [*владелец*.]*имя_представления*] ...

Параметры

имя_представления

Имя представления, которое нужно удалить. Чтобы удалить представление с таким же именем, принадлежащее другому пользователю в текущей базе данных, перед именем представления необходимо указать имя владельца. По умолчанию *владелец* – текущий пользователь.

Примеры

Удаление из текущей базы данных представления new_price:

```
drop view new_price
```

Использование

- Команда drop view удаляет определение представления и другую связанную с ним информацию, включая привилегии, из системных таблиц sysobjects, syscolumns, syscomments, sysdepends, sysprocedures и sysprotects.
- Проверка того, что представление существует, выполняется при каждом запросе к представлению, сделанным, например, другим представлением или хранимой процедурой.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Полномочия на удаление представления с помощью команды drop view по умолчанию принадлежат владельцу представления и не могут передаваться.

См. также

Команды create view

Системные процедуры sp_depends, sp_help, sp_helptext

dump database

Описание

Создает резервную копию всей базы данных, включая журнал транзакций, в формате, который можно прочитать с помощью команды load database. Резервное копирование и загрузка выполняются через сервер Backup Server.

Синтаксис

```
dump database имя_базы_данных
        to [compress::[уровень_сжатия::]]устройство
            [at имя_сервера_backup_server]
            [density = значение_плотности,
            blocksize = число_байтов,
            capacity = число_килобайтов,
            dumpvolume = имя_тома,
            file = имя_файла]
        [stripe on [compress::[уровень_сжатия::]]устройство
            [at имя_сервера_backup_server]
            [density = значение_плотности,
            blocksize = число_байтов,
            capacity = число_килобайтов,
            dumpvolume = имя_тома,
            file = имя_файла]
        [[stripe on [compress::[уровень_сжатия::]]устройство
            [at имя_сервера_backup_server]
            [density = значение_плотности,
            blocksize = число_байтов,
            capacity = число_килобайтов,
            dumpvolume = имя_тома,
            file = имя_файла]
        [with {
            density = значение_плотности
            blocksize = число_байтов,
            capacity = число_килобайтов,
            dumpvolume = имя_тома,
            file = имя_файла
            [dismount | nodismount],
            [nounload | unload],
            retaindays = число_дней
            [noinit | init],
            notify = {client | operator_console}
            } ]
```

Параметры

имя_базы данных

Имя базы данных, из которой копируются данные. Может быть литералом, локальной переменной или параметром хранимой процедуры.

compress::уровень_сжатия

Число от 0 до 9, при этом 0 означает отсутствие сжатия, а 9 – наибольший уровень сжатия. Если *уровень_сжатия* не указан, то значение по умолчанию равно 1. Дополнительную информацию о параметре compress см. в главе 27, “Резервное копирование и восстановление пользовательских баз данных”, книги *Руководство по системному администрированию*.

Примечание. Параметр compress работает только с локальными архивами; вместе с ним нельзя указывать параметр *имя_сервера_backup_server*.

---

to устройство

Устройство, на которое будут скопированы данные. Дополнительную информацию о формате, который должен использоваться при указании устройства для хранения резервных копий, см. в подразделе “Указание устройств для хранения резервных копий” этого раздела.

at имя_сервера_backup_server

Имя сервера Backup Server. Чтобы выполнить резервное копирование на сервер Backup Server по умолчанию, этот параметр не нужно указывать. Его необходимо указывать только при резервном копировании по сети на удаленный сервер Backup Server. В этом параметре можно задать до 32 удаленных серверов Backup Server. При резервном копировании по сети нужно указать *сетевое имя* для удаленного сервера Backup Server, работающего на машине, к которой подключено устройство для хранения резервных копий. Если на платформе используются файлы интерфейсов, то *имя_сервера_backup_server* должно находиться в файле интерфейсов.

density = значение_плотности

Переопределяет плотность по умолчанию для ленточного устройства. Допустимыми значениями плотности являются 800, 1600, 6250, 6666, 10000 и 38000. Не все эти значения допустимы для каждого ленточного устройства, поэтому необходимо уточнить, какие значения плотности допустимы для используемого устройства.

blocksize = число_байтов

Переопределяет стандартный размер блока для устройства для хранения резервных копий. Размер блока должен быть не меньше одной страницы базы данных (2048 байтов для большинства систем) и кратен размеру страницы базы данных. Для достижения оптимальной производительности размер блока blocksize должен быть степенью 2 (например, 65536, 131072 или 262144).

capacity = число_килобайтов

Максимальный объем данных, который можно записать на один том ленточного устройства. Емкость должна быть не меньше пяти страниц базы данных и не должна превышать рекомендуемое значение для устройства.

Обычно указывают емкость, равную 70 процентам максимальной емкости устройства; 30 процентов оставляют для интервалов между записями и для служебной информации, например маркеров ленты. Максимальная емкость – это емкость логической области на ленточном устройстве, а не емкость самого устройства. Это правило работает в большинстве случаев, но не всегда, поскольку для устройств разных изготовителей могут потребоваться различные объемы служебной области.

В операционной системе UNIX, которая не может точно обнаруживать маркер окончания ленточного устройства, необходимо указать объем резервируемых данных в килобайтах. Параметр capacity *необходимо* задать, если устройство для хранения резервных копий было указано с помощью физического пути. Если в качестве устройства для хранения резервных копий было задано имя логического устройства и не была указана емкость, то сервер Backup Server будет использовать параметр size, хранящийся в системной таблице sysdevices.

dumpvolume = имя_тома

Устанавливает имя, которое будет присвоено тому. Максимальная длина *имени_тома* составляет 6 знаков. Сервер Backup Server записывает переменную *имя_тома* в метку ANSI ленточного устройства при перезаписи существующей резервной копии, резервном копировании на совершенно новую ленту или резервном копировании на ленту, содержимое которой нельзя распознать. Команда load database проверяет метку и выдает сообщение об ошибке, если загружается неправильный том.

Предупреждение. Каждый том ленточного устройства должен быть помечен при создании резервной копии, чтобы оператор знал, какой том нужно загрузить.

---

stripe on устройство

Дополнительное устройство для хранения резервных копий. Можно указать до 32 устройств, включая *устройство*, заданное в инструкции to. Сервер Backup Server расщепляет базу данных на приблизительно равные части и посылает их на различные устройства. Резервное копирование выполняется одновременно на все устройства, что сокращает затраты времени и объем вносимых в тома изменений в ходе резерв-

ного копирования. Информацию о том, как указывать устройство для хранения резервных копий, см. в разделе “Указание устройств для хранения резервных копий” на стр. 527.

dismount | nodismount
На платформах, поддерживающих логический демонтаж, указывает, будут ли демонтированы ленты. По умолчанию все ленты, использованные для резервного копирования, демонтируются после его завершения. Чтобы ленты оставались доступными для последующих операций резервного копирования и загрузки, нужно указать nodismount.

nounload | unload
Указывает, будут ли перематываться ленты после резервного копирования. По умолчанию ленты не перематываются, чтобы можно было сделать дополнительные резервные копии на тот же том ленты. Параметр unload нужно указать для последнего файла резервной копии, добавляемого в том, состоящий из нескольких резервных копий. В результате лента будет перемотана и выгружена после выполнения резервного копирования.

retaindays = число_дней
В операционных системах UNIX – при резервном копировании на диск задает количество дней, в течение которых Backup Server защищает резервную копию от перезаписи. Если попытаться перезаписать резервную копию до того, как истек срок ее действия, сервер Backup Server запросит подтверждение, прежде чем совершить перезапись.

Примечание. Этот параметр применим только при резервном копировании на диск. Он не имеет смысла, если резервное копирование происходит на ленту.

число_дней должно быть положительным целым числом (или 0 для резервных копий, которые можно сразу же перезаписать). Если не указать значение параметра retaindays, то сервер Backup Server будет использовать значение параметра tape retention in days, устанавливаемое системной процедурой sp_configure.

noinit | init
Определяет, будет ли резервная копия добавлена к существующим резервным копиям или том ленты будет инициализирован заново (перезаписан). По умолчанию сервер Adaptive Server добавляет резервные копии за последним маркером окончания ленты, что позволяет резервировать дополнительные базы данных в этот же том. Новые резервные копии можно добавлять только к последнему тому многотом-

ной резервной копии. Параметр init нужно указать для первой базы данных, резервируемой на ленточное устройство, чтобы перезаписать содержимое ленты.

Параметр init нужно указать, чтобы сервер Backup Server сохранил или обновил характеристики ленточного устройства в файле конфигурации ленты. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

file = имя_файла
Имя файла резервной копии. Имя не может быть длиннее 17 символов и должно соответствовать правилам операционной системы. Дополнительную информацию см. в разделе "Файлы резервных копий" на стр. 528.

notify = {client | operator_console}
Переопределяет получателя сообщения, заданного по умолчанию.

В операционных системах, имеющих функцию терминалов оператора, сообщения об изменении тома всегда посылаются терминалу оператора на компьютере, на котором работает сервер Backup Server. Если указать client, то остальные сообщения сервера Backup Server будут направляться в сеанс терминала, инициировавший команду dump database.

В операционных системах, в которых нет функции терминалов оператора (например, UNIX), сообщения посылаются клиенту, который инициировал команду dump database. Если указать operator_console, то сообщения будут направляться на терминал, на котором работает сервер Backup Server.

Примеры

Пример 1. Резервное копирование базы данных pubs2 на ленточное устройство. Если у ленты есть метка ANSI, то эта команда добавляет эту резервную копию к содержащимся на ленте файлам, поскольку параметр init не указан:

```
dump database pubs2
    to "/dev/nrmt0"
```

Пример 2. *(Для операционных систем UNIX.)* Резервное копирование базы данных pubs2 на сервер Backup Server с именем REMOTE_BKP_SERVER. В этой команде указано три устройства для хранения резервных копий, поэтому сервер Backup Server сохраняет на каждом устройстве резервные копии, содержащие примерно по одной трети базы данных. Эта команда добавляет резервную копию к существующим

файлам на ленточных устройствах. Для операционной системы UNIX параметр retaindays указывает, что ленты не могут быть перезаписаны в течение 14 дней:

```
dump database pubs2
    to "/dev/rmt4" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt0" at REMOTE_BKP_SERVER
with retaindays = 14
```

Пример 3. Параметр init инициализирует том ленты, перезаписывая все находящиеся на ней файлы:

```
dump database pubs2
    to "/dev/nrmt0"
    with init
```

Пример 4. Перемотка тома резервной копии после завершения резервного копирования:

```
dump database pubs2
    to "/dev/nrmt0"
    with unload
```

Пример 5. (*Для операционных систем UNIX*.) Если указана инструкция notify, то сообщения сервера Backup Server, запрашивающие изменение тома, посылаются клиенту, сделавшему запрос на резервное копирование, а не консоли компьютера с сервером Backup Server (являющейся получателем по умолчанию):

```
dump database pubs2
    to "/dev/nrmt0"
    with notify = client
```

Пример 6. Создание сжатой резервной копии (с уровнем сжатия 4) базы данных pubs2 в файле *dmp090100.dmp*:

```
dump database pubs2 to
   "compress::4::/opt/bin/Sybase/dumps/dmp090100.dmp"
```

Использование

- В таблице 7-22 приведено описание команд и системных процедур, использующихся для резервного копирования баз данных.

Таблица 7-22. Команды, используемые для резервного копирования баз данных и журналов

Операция	Команды, реализующие операцию
Обычное резервное копирование всей базы данных, включая журнал транзакций	dump database
Обычное резервное копирование журнала транзакций с последующей очисткой неактивной части	dump transaction

Операция	Команды, реализующие операцию
Резервное копирование журнала транзакций после сбоя устройства хранения базы данных	dump transaction with no_truncate
Очистка журнала без создания резервной копии, а затем копирование всей базы данных	dump transaction with truncate_only dump database
Очистка журнала в случае сбоя обычно используемого метода (из-за недостатка пространства в журнале) с последующим копированием всей базы данных	dump transaction with no_log dump database
Ответ на сообщения Backup Server о смене тома	sp_volchanged

Ограничения

- Нельзя выполнять резервное копирование из сервера 11.x Adaptive Server в сервер 10.x Backup Server.
- Нельзя размещать резервные копии базы данных Sybase и данные, не относящиеся к Sybase (например, архивов UNIX), на одном ленточном устройстве.
- Если база данных участвует в межбазовом ограничении ссылочной целостности, то в системной таблице sysreferences хранится *имя* внешней базы данных, а не ее идентификационный номер. Сервер Adaptive Server не гарантирует ссылочной целостности, если база данных переименована или загружена на другой сервер с помощью команды load database.

  Предупреждение. Если резервная копия создается для того, чтобы загрузить базу данных под другим именем или переместить ее на другой сервер Adaptive Server, то перед резервным копированием нужно удалить все внешние ограничения ссылочной целостности с помощью команды alter table.

- Команду dump database нельзя использовать в пользовательской транзакции.
- Если выполнить команду dump database для базы данных, в которой уже выполняется команда dump transaction, то команда dump database не будет выполняться, пока команда dump transaction не завершится.
- На ленте с картриджем 1/4 дюйма можно сохранять не более одной резервной копии базы данных или журнала транзакции.
- Нельзя резервировать базу данных, если у нее есть автономные страницы. Чтобы принудительно перевести автономные страницы в оперативный режим, выполните системную процедуру sp_forceonline_db или sp_forceonline_page.

Планирование резервного копирования

- Резервное копирование баз данных сервера Adaptive Server является *динамическим* – оно может выполняться при активной базе данных. Однако это несколько замедляет работу системы, поэтому команду dump database лучше выполнять, когда обновления в базе данных происходят не слишком интенсивно.
- *Базу данных* master *нужно резервировать регулярно и часто*. Кроме того, резервную копию базы данных master нужно создавать после каждого выполнения команд create database, alter database и disk init.
- Резервную копию базы данных model нужно создавать при каждом изменении в базе данных.
- После создания базы данных нужно сразу же сделать копию всей базы данных с помощью команды dump database. В новой базе данных нельзя выполнять команду dump transaction до выполнения команды dump database.
- При каждом добавлении или удалении межбазового ограничения ссылочной целостности или удалении таблицы, содержащей межбазовое ограничение, необходимо выполнить резервное копирование *обеих* измененных баз данных.

  Предупреждение. Загрузка более ранних резервных копий этих баз данных может привести к повреждениям данных.

- Должен быть разработан график регулярного резервного копирования пользовательских баз данных и их журналов транзакций.
- Использование порогов позволяет автоматизировать процедуры резервного копирования. Чтобы воспользоваться последним порогом сервера Adaptive Server, отделите сегменты журнала пользовательских баз данных от сегментов данных (то есть храните их на разных устройствах). Дополнительную информацию о порогах см. в книге *Руководство по системному администрированию*.

Резервное копирование системных баз данных

- Базы данных master, model и sybsystemprocs не имеют отдельных сегментов для журналов транзакций. Для них нужно сначала очистить журнал с помощью команды dump transaction with truncate_only, а затем создать резервную копию командой dump database.

- Резервные копии базы данных master необходимы для восстановления в случае сбоев, затрагивающих эту базу данных. Пошаговые инструкции по созданию резервной копии и восстановлению базы данных master см. в книге *Руководство по системному администрированию*.
- Если резервные копии создаются на съемных носителях, то база данных master должна полностью помещаться на одном томе (если нет другого сервера Adaptive Server, который может реагировать на сообщения о смене тома).

Указание устройств для хранения резервных копий

- Устройство для хранения резервных копий можно задать с помощью литерала, локальной переменной или параметра хранимой процедуры.
- Нельзя делать резервное копирование на устройство NULL (в UNIX таким устройством является */dev/null*).
- При резервном копировании на ленточные устройства и диски можно расщеплять резервную копию. Помещать несколько резервных копий на одно устройство можно только при резервном копировании на ленточные устройства.
- Можно указать локальное устройство для хранения резервных копий следующим образом:
  - как имя логического устройства из системной таблицы sysdevices;
  - указав полный путь;
  - указав относительный путь.

  В сервере Backup Server можно указывать относительный путь (относительно текущего рабочего каталога сервера Adaptive Server).
- При резервном копировании по сети нужно указать полный путь устройства для хранения резервных копий. Путь должен быть допустимым для машины, на которой работает сервер Backup Server. Если имя включает какие-либо символы помимо букв, цифр и символа подчеркивания (_), его необходимо заключить в кавычки.
- При выполнении команды dump могут возникнуть проблемы, связанные с владением и полномочиями на устройство для хранения резервных копий. Системная процедура sp_addumpdevice добавляет устройство в системную таблицу, но не гарантирует того, что можно будет выполнить резервное копирование на это устройство или создать файл как устройство для хранения резервных копий.

- Можно выполнять несколько операций резервного копирования (или загрузок) одновременно, но каждая операция должна работать со своим устройством для хранения резервных копий.
- Если файл устройства уже существует, то сервер Backup Server перезаписывает его, но не очищает. Например, допустим, что размер файла, в который была помещена резервная копия базы данных, равен 10 МБ. Если затем записать в этот файл другую резервную копию базы данных, имеющую меньший размер, то размер файла по-прежнему будет равен 10 МБ.

Определение характеристик ленточного устройства

- Если в команде dump не указан параметр init и сервер Backup Server не может определить тип устройства, то эта команда не будет выполнена. Дополнительную информацию можно найти в книге *Руководство по системному администрированию*.

Серверы Backup Server

- Сервер Backup Server должен работать на той же машине, что и сервер Adaptive Server. Запись о сервере Backup Server должна содержаться в таблице master..sysservers. Эта запись создается в процессе установки системы или перехода на новую версию, и ее нельзя удалять.
- Если устройства для хранения резервных копий размещены на другой машине и резервное копирование выполняется через сеть, то необходимо, чтобы на удаленной машине также был установлен сервер Backup Server.

Файлы резервных копий

- Выполнение резервного копирования базы данных с параметром init перезаписывает любые существующие файлы на ленте или диске.
- Если создается несколько резервных копий на ленточном устройстве и для каких-либо двух резервных копий указываются одинаковые имена файлов (в параметре FILENAME), сервер Adaptive Server добавляет вторую резервную копию к архивному устройству. При этом со второй резервной копии нельзя будет провести восстановление, поскольку по указанному имени сервер Adaptive Server обнаружит первый экземпляр образа резервной копии и проведет восстановление с него. Сервер Adaptive Server не будет искать другие образы резервных копий с тем же именем.
- Сервер Backup Server отправляет имя файла резервной копии получателю, указанному в инструкции with notify. На ленточном устройстве с резервной копией желательно указать имя базы данных, имя файла, дату резервного копирования и другую важную информацию.

Если лента, с которой загружается база данных, не имеет такой опознавательной метки, нужно указать параметры with headeronly и with listonly, чтобы можно было определить содержимое ленты.

Имена файлов и имена архивов

- Имя файла идентифицирует базу данных, чьей резервной копией этот файл является, и дату резервного копирования. Однако синтаксис *имени_файла* трактуется по-разному в зависимости от того, делается ли резервное копирование на диск или на ленту UNIX:

  ```
  file = имя_файла
  ```

  При резервном копировании на диск путь также является именем файла.

  При резервном копировании на ленту UNIX путь не является именем файла. В стандартном формате ANSI для File Interchange метка HDR1 содержит поле с именем файла. Для лент, соответствующих спецификации ANSI, это поле в метке идентифицирует имя файла. В спецификации ANSI эти метки могут использоваться только для ленточных устройств, но не для файлов на диске.

  В связи с этим возникают две проблемы:

  - Имена файлов на ленточных устройствах в операционной системе UNIX не соответствуют правилам ANSI. UNIX воспринимает данные на ленте как непомеченные. Хотя данные могут быть разделены на файлы, у этих файлов нет имени.
  - В сервере Backup Server метки ANSI для ленточного устройства содержат информацию об архиве, что не соответствует стандарту ANSI. Поэтому файлы на диске также имеют метки ANSI, поскольку там хранится имя архива.

  Значение имени файла зависит от типа выполняемого резервного копирования. Например, в следующем синтаксисе:

```
dump database имя_базы_данных to 'имя_файла' with file='имя_файла'
```

  - Первое *имя_файла* указывает имя пути, используемое для отображения файла.
  - Второе *имя_файла* – это имя архива (хранящееся в метке HDR1 в архиве). Пользователь может указать это имя с помощью параметра file=имя_файла команды dump или load.

  Если имя архива указано, то сервер использует это имя при загрузке базы данных, чтобы найти выбранный архив.

  Если имя архива не указано, то сервер загружает первый архив, который он находит.

В обоих случаях параметр file='имя_архива' задает имя, которое будет сохранено в метке HDR1 и использоваться последующей командой load для проверки того, что она работает с правильными данными.

Если имя архива не указано, оно будет создано командой dump; команда load в этом случае будет использовать первое имя файла, которое она встречает.

Значение параметра *имя_файла* в инструкции to '*имя_файла*' зависит от того, на каком носителе находится резервная копия – на диске или на ленте:

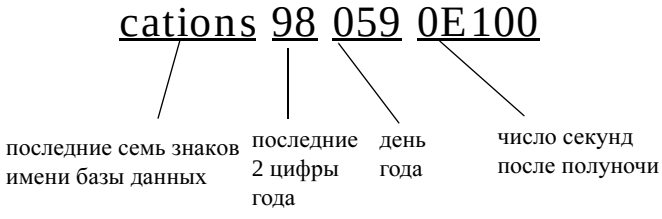
- Если резервная копия находится на ленте, то '*имя_файла*' – это имя ленточного устройства;
- Если резервная копия находится на диске, то это имя файла на диске.

Если резервная копия находится на диске, а '*имя_файла*' не содержит полный путь, то к нему дописывается текущий рабочий каталог сервера.

- Если резервное копирование выполняется на ленту, а имя файла не указано, то сервер Backup Server создает стандартное имя файла за счет конкатенации следующих строк:
  - Последних семи символов имени базы данных
  - Двух цифр, обозначающих год
  - Трех цифр, обозначающих день года (от 1 до 366)
  - Времени создания файла резервной копии в шестнадцатеричном формате

  Например, файл *cations980590E100* содержит копию базы данных publications, созданную в пятьдесят девятый день 1998 года.

Рис. 7-2. Правило именования резервных копий базы данных, находящихся на ленте



Имена томов

- Метки томов резервных копий соответствуют стандарту ANSI по меткам для лент. Метка содержит номер логического тома и позицию устройства в наборе резервных копий.
- В процессе загрузки сервер Backup Server использует метку ленты для проверки того, что тома смонтированы в правильном порядке. Это позволяет загружать данные с меньшего числа устройств по сравнению с тем, которое было использовано при резервном копировании.

Примечание. Когда резервное копирование или загрузка происходит через сеть, для всех операций необходимо указать одно и то же число устройств для хранения резервных копий.

Изменение томов резервной копии

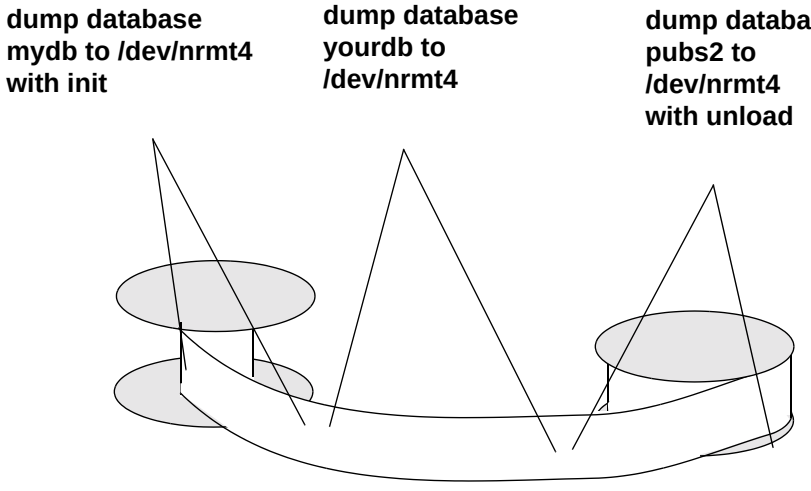
- *В операционной системе UNIX* сервер Backup Server просит сменить том при заполнении ленточного устройства. Когда другой том смонтирован, нужно уведомить об этом сервер Backup Server, выполнив системную процедуру sp_volchanged на любом сервере Adaptive Server, взаимодействующем с сервером Backup Server.
- Если сервер Backup Server обнаружил неполадку в текущем смонтированном томе, то он посылает клиенту или консоли оператора требование о смене тома. При получении этого сообщения нужно выполнить системную процедуру sp_volchanged.

Добавление к тому и перезапись тома

- По умолчанию сервер Backup Server использует параметр noinit, то есть записывает последовательные резервные копии на один и тот же том ленты, позволяя эффективно использовать ленточные носители с большим объемом пространства. Данные добавляются за последним маркером окончания ленты. Новые резервные копии можно добавлять только к последнему тому многотомной резервной копии. Прежде чем записывать на ленту, сервер Backup Server проверяет, не истек ли срок действия первого файла. Если лента содержит данные, которые не относятся к Sybase, то сервер не производит на нее запись, чтобы не удалять потенциально ценную информацию.
- Параметр init указывает, что том будет инициализирован заново. Если указывается параметр init, сервер Backup Server перезаписывает любую существующую информацию, даже если лента содержит данные, не относящиеся к Sybase, или срок первого файла еще не истек, или лента имеет ограничения доступа ANSI.

- На Рис. 7-3 показано резервное копирование трех баз данных на один том со следующими параметрами:
  - init – для инициализации ленты для первой резервной копии;
  - noinit (по умолчанию) – для добавления последовательных резервных копий;
  - unload – для перемотки и выгрузки ленты после последней операции резервного копирования

Рис. 7-3. Резервное копирование нескольких баз данных на один том



Резервное копирование с 32-разрядной операционной системы на 64-разрядную

Резервные копии базы данных в 32-разрядной версии сервера Adaptive Server полностью совместимы с 64-разрядной версией сервера Adaptive Server, работающей на той же платформе, и наоборот.

Резервное копирование баз данных с зеркальными копиями

- В начале выполнения команды dump database сервер Adaptive Server передает серверу Backup Server имя устройства, являющегося основным по отношению ко всем устройствам для хранения базы данных и журналов. Если для первичного устройства отключен режим зеркального копирования, сервер Adaptive Server передает вместо него имя вторичного устройства. Если с каким-либо именованным устройством происходит сбой до того, как сервер Backup Server завершил передачу данных, то сервер Adaptive Server аварийно завершает резервное копирование.

- Если пользователь пытается отключить какое-либо именованное устройство базы данных в процессе выполнения команды dump database, то сервер Adaptive Server выводит соответствующее сообщение. Пользователь, выполняющий команду disk unmirror, может принудительно завершить резервное копирование или отложить выполнение команды disk unmirror до тех пор, пока резервное копирование не будет завершено.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду dump database могут выполнять только системный администратор, владелец базы данных и пользователи с ролью Operator.

См. также

Команды dump transaction, load database, load transaction

Системные процедуры sp_addthreshold, sp_addumpdevice, sp_dropdevice, sp_dropthreshold, sp_helpdevice, sp_helpdb, sp_helpthreshold, sp_logdevice, sp_spaceused, sp_volchanged

dump transaction

Описание

Создает копию журнала транзакций и удаляет неактивную часть.

Синтаксис

Создание обычной резервной копии журнала:

```
dump tran[saction] имя_базы_данных
    to [compress::[уровень_сжатия::]]устройство
        [at имя_сервера_backup_server]
        [density = значение_плотности,
        blocksize = число_байтов,
        capacity = число_килобайтов,
        dumpvolume = имя_тома,
        file = имя_файла]
    [stripe on [compress::[уровень_сжатия::]]устройство
        [at имя_сервера_backup_server]
        [density = значение_плотности,
        blocksize = число_байтов,
        capacity = число_килобайтов,
        dumpvolume = имя_тома,
        file = имя_файла]
    [[stripe on [compress::[уровень_сжатия::]]устройство
        [at имя_сервера_backup_server]
        [density = значение_плотности,
        blocksize = число_байтов,
        capacity = число_килобайтов,
        dumpvolume = имя_тома,
        file = имя_файла]
    [with {
        density = значение_плотности
        blocksize = число_байтов,
        capacity = число_килобайтов,
        dumpvolume = имя_тома,
        file = file_name
        [dismount | nodismount],
        [nounload | unload],
        retaindays = число_дней,
        [noinit | init],
        notify = {client | operator_console},
        standby_access }]
```

Очистка журнала без создания резервной копии:

```
dump tran[saction] имя_базы_данных
    with truncate_only
```

Очистка целиком заполненного журнала *(используется только в крайнем случае)*:

```
dump tran[saction] имя_базы_данных
    with no_log
```

Создание резервной копии журнала после сбоя устройства для хранения базы данных:

```
dump tran[saction] имя_базы_данных
    to [compress::[уровень_сжатия::]]устройство
        [at имя_сервера_backup_server]
        [density = значение_плотности,
        blocksize = число_байтов,
        capacity = число_килобайтов,
        dumpvolume = имя_тома,
        file = имя_файла]
    [stripe on [compress::[уровень_сжатия::]]устройство
        [at имя_сервера_backup_server]
        [density = значение_плотности,
        blocksize = число_байтов,
        capacity = число_килобайтов,
        dumpvolume = имя_тома,
        file = имя_файла]
    [[stripe on [compress::[уровень_сжатия::]]устройство
        [at имя_сервера_backup_server]
        [density = значение_плотности,
        blocksize = число_байтов,
        capacity = число_килобайтов,
        dumpvolume = имя_тома,
        file = имя_файла]
    [with {
        density = значение_плотности
        blocksize = число_байтов,
        capacity = число_килобайтов,
        dumpvolume = имя_тома,
        file = имя_файла
        [dismount | nodismount],
        [nounload | unload],
        retaindays = число_дней
        [noinit | init],
        no_truncate,
        notify = {client | operator_console}}]
```

Параметры

имя_базы данных

Имя базы данных, из которой копируются данные. Может быть литералом, локальной переменной или параметром хранимой процедуры.

compress::уровень_сжатия

Число от 0 до 9, при этом 0 означает отсутствие сжатия, а 9 – наибольший уровень сжатия. Если *уровень_сжатия* не указан, то значение по умолчанию равно 1. Дополнительную информацию о параметре

compress см. в главе 27, “Резервное копирование и восстановление пользовательских баз данных”, в книге *Руководство по системному администрированию*.

Примечание. Параметр compress работает только с локальными архивами; вместе с ним нельзя указывать параметр *имя_сервера_backup_server*.

---

truncate_only
: Удаляет неактивную часть журнала *без создания резервной копии*. Используется в базах данных, у которых нет сегментов журналов на отдельном от сегментов данных устройстве. Вместе с этим параметром не нужно указывать устройство для хранения резервных копий или имя сервера Backup Server.

no_log
: Удаляет неактивную часть журнала *без создания резервной копии и без записи этой процедуры в журнал транзакций*. Параметр no_log используется, только когда пространство журнала заполнено целиком и нельзя выполнить обычную команду dump transaction. Этот параметр можно указывать только в крайнем случае и только один раз после того, как произошла ошибка при выполнении команды dump transaction with truncate_only. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

to устройство
: Устройство, на которое выполняется резервное копирование данных. Дополнительную информацию о формате, который должен использоваться при указании устройства для хранения резервных копий, см. в подразделе “Указание устройств для хранения резервных копий” на стр. 527.

at имя_сервера_backup_server
: Имя сервера Backup Server. Чтобы выполнить резервное копирование на сервер Backup Server по умолчанию, этот параметр не нужно указывать. Этот параметр необходимо указывать только при резервном копировании по сети на удаленный сервер Backup Server. В этом параметре можно задать до 32 удаленных серверов Backup Server. При резервном копировании по сети нужно указать *сетевое имя* для удаленного сервера Backup Server, работающего на машине, к которой подключено устройство для хранения резервных копий. Если операционная система использует файлы интерфейсов, то *имя_сервера_backup_server* должно находиться в файле интерфейсов.

density = значение_плотности
: Переопределяет плотность по умолчанию для ленточного устройства. Допустимыми значениями плотности являются 800, 1600, 6250, 6666, 10000 и 38000. Не все эти значения допустимы для каждого ленточного устройства, поэтому необходимо уточнить, какие значения плотности допустимы для используемого устройства.

blocksize = число_байтов
: Переопределяет размер блока по умолчанию для устройства для хранения резервных копий. Размер блока должен быть не меньше одной страницы базы данных (2048 байтов в большинстве систем) и кратен размеру страницы базы данных.

Примечание. По возможности должен использоваться размер блока по умолчанию (поскольку это оптимальный размер блока для системы).

---

capacity = число_килобайтов
: Максимальный объем данных, который может быть записан на один ленточный том. Емкость должна быть не меньше пяти страниц базы данных и не должна превышать рекомендуемое значение для устройства.

  Обычно указывают емкость, равную 70 процентам максимальной емкости устройства; 30 процентов оставляют для интервалов между записями и для служебной информации, например маркеров ленты. Это правило работает в большинстве случаев, но не всегда, поскольку для устройств разных изготовителей могут потребоваться различные объемы служебной области.

  В операционной системе UNIX, которая не может точно обнаруживать маркер окончания ленточного устройства, необходимо указать объем резервируемых данных в килобайтах. Параметр capacity *необходимо* задать, если устройство для хранения резервных копий было указано с помощью физического пути. Если в качестве устройства для хранения резервных копий было задано имя логического устройства и не была указана емкость, то сервер Backup Server будет использовать параметр size, хранящийся в системной таблице sysdevices.

dumpvolume = имя_тома
: Устанавливает имя, которое будет присвоено тому. Максимальная длина *имени_тома* составляет 6 знаков. Сервер Backup Server записывает переменную *имя_тома* в метку ANSI ленточного устройства при перезаписи существующей резервной копии, резервном копировании на совершенно новую ленту или резервном копировании на

ленту, содержимое которой нельзя распознать. Команда load transaction проверяет метку и выдает сообщение об ошибке, если загружается неправильный том.

stripe on устройство
: Дополнительное устройство для хранения резервных копий. Можно указать до 32 устройств, включая *устройство*, заданное в инструкции to. Сервер Backup Server расщепляет журнал на приблизительно равные части и посылает их на различные устройства. Создание резервных копий на разных устройствах происходит одновременно, что экономит время и снижает количество смен томов, требуемое для операции. Информацию о том, как указать устройство для хранения резервных копий, см. в разделе "Указание устройств для хранения резервных копий" на стр. 527.

dismount | nodismount
: *Для платформ, поддерживающих логический демонтаж*, – определяет, остаются ли ленты смонтированными. По умолчанию все ленты, используемые для резервного копирования, демонтируются после его завершения. Для того чтобы ленты оставались доступными для последующих операций резервного копирования и загрузки, нужно указать nodismount.

nounload | unload
: Указывает, будут ли перематываться ленты после резервного копирования. По умолчанию ленты не перематываются, что позволяет создавать дополнительные резервные копии на этом же томе ленты. Параметр unload нужно указать для последнего файла резервной копии, добавляемого в том, состоящий из нескольких резервных копий. В результате лента будет перемотана и выгружена после выполнения резервного копирования.

retaindays = число_дней
: *Для платформ UNIX* – задает число дней, в течение которых сервер Backup Server обеспечивает защиту резервной копии от перезаписи. Если попытаться перезаписать резервную копию до того, как истек срок ее действия, сервер Backup Server запрашивает подтверждение, прежде чем совершить перезапись.

Примечание. Этот параметр применим только при резервном копировании на диски, 1/4-дюймовые картриджи и носители, которые могут содержать только один файл. Для резервного копирования на носители, которые могут содержать более одного файла, этот параметр применим ко всем томам, кроме первого.

Число_дней должно быть положительным целым числом (или 0 для резервных копий, которые можно сразу же перезаписать). Если не указать значение параметра retaindays, то сервер Backup Server будет использовать значение параметра tape retention in days (параметра уровня сервера), устанавливаемое системной процедурой sp_configure.

noinit | init

Указывает, будет ли резервная копия добавлена к существующим резервным копиям или том ленты будет инициализирован заново (то есть перезаписан). По умолчанию сервер Adaptive Server добавляет резервные копии за последним маркером окончания ленты, что позволяет резервировать дополнительные базы данных в этот же том. Новые резервные копии можно добавлять только к последнему тому многотомной резервной копии. Параметр init нужно указать для первой базы данных, резервируемой на ленточное устройство, чтобы перезаписать содержимое ленты.

Параметр init нужно указать, чтобы сервер Backup Server сохранил или обновил характеристики ленточного устройства в файле конфигурации ленты. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

file = имя_файла

Имя файла резервной копии. Имя не может быть длиннее 17 символов и должно соответствовать правилам именования файлов операционной системы. Если имя файла не указывается, то сервер Backup Server создаст файл с именем по умолчанию. Дополнительную информацию см. в разделе “Файлы резервных копий” на стр. 528.

no_truncate

Выполняет резервное копирование журнала транзакций (используя указатель на журнал транзакций в базе данных master), *даже если диск, содержащий сегменты данных базы данных, недоступен*. Установка with no_truncate обеспечивает восстановление из журнала на текущий момент времени, если журнал транзакций находится на неповрежденном устройстве, а база данных master и пользовательские базы данных находятся на разных физических устройствах.

notify = {client | operator_console}

Переопределяет адресата сообщения, заданного по умолчанию.

- В операционных системах, имеющих функцию терминалов оператора, сообщения о смене тома всегда посылаются терминалу оператора на компьютере, на котором работает сервер Backup Server. Если указать client, то остальные сообщения сервера Backup Server будут направляться в сеанс терминала, инициировавший команду dump transaction.

- В операционных системах, в которых нет терминалов оператора (например, UNIX), сообщения посылаются клиенту, который инициировал команду dump transaction. Если указать operator_console, то сообщения будут направляться на терминал, на котором работает сервер Backup Server.

with standby_access
Указывает, что резервное копирование должно выполняться только для завершенных транзакций. Резервное копирование закончится тогда, когда будет создана резервная копия последней журнальной записи в самой последней завершенной транзакции, при которой не было открытых транзакций.

Примеры

Пример 1. Создание резервной копии журнала транзакций на ленте (новая копия добавляется к существующим файлам на ленте, поскольку параметр init не указан):

```
dump transaction pubs2
    to "/dev/nrmt0"
```

Пример 2. Создание резервной копии журнала транзакций для базы данных mydb на сервере Backup Server с именем REMOTE_BKP_SERVER. Backup Server резервирует приблизительно по половине журнала на каждое из двух устройств. Параметр init указывает, что все существующие файлы на ленте будут перезаписаны, а параметр retaindays – что ленты не могут быть перезаписаны в течение 14 дней:

```
dump transaction mydb
    to "/dev/nrmt4" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
with init, retaindays = 14
```

Пример 3. Резервное копирование завершенных транзакций из журнала транзакций inventory_db на устройство dev1:

```
dump tran inventory_db to dev1 with standby_access
```

Использование

- В таблице 7-23 приведено описание команд и системных процедур, используемых для резервного копирования баз данных и журналов.

Таблица 7-23. Команды, используемые для резервного копирования баз данных и журналов

Выполняемая операция	Команды, реализующие операцию
Обычное резервное копирование всей базы данных, включая журнал транзакций	dump database
Обычное резерное копирование журнала транзакций с последующей очисткой неактивной части	dump transaction

Выполняемая операция	Команды, реализующие операцию
Резервное копирование журнала транзакций после сбоя устройства хранения базы данных	dump transaction with no_truncate
Очистка журнала без создания резервной копии, а затем копирование всей базы данных	dump transaction with truncate_only dump database
Очистка журнала после того, как при выполнении обычного метода возникла ошибка из-за нехватки места в журнале, а затем копирование всей базы данных	dump transaction with no_log dump database
Ответ на сообщения Backup Server о смене тома	sp_volchanged

Ограничения

- Нельзя делать резервное копирование на устройство NULL (*/dev/null* в UNIX).
- Нельзя выполнять команду dump transaction в транзакции.
- На ленте с картриджем 1/4 дюйма можно сохранять не более одной резервной копии базы данных или журнала транзакций.
- Нельзя выполнять резервное копирование журнала транзакций, пока включен параметр trunc log on chkpt, а также после включения параметра select into/bulk copy/pllsort и выполнения над базой данных минимально регистрируемых операций select into, ускоренного массового копирования, операций writetext (не регистрируемых по умолчанию) или параллельной сортировки. В таких случаях нужно использовать команду dump database.

  Предупреждение. Нельзя изменять таблицу журнала syslogs с помощью команд delete, update и insert.

- Если журнальные сегменты базы данных хранятся на том же устройстве, что и сегменты данных, то нельзя использовать команду dump transaction для копирования и очистки журнала.
- Если пользователь или процедура последнего порога выполняет команду dump transaction для базы данных, в которой выполняется команда dump database или другая команда dump transaction, то новая команда ждет завершения уже выполняющейся.
- Для восстановления базы данных выполните команду load database для загрузки самой последней резервной копии базы данных, а затем команду load transaction для загрузки каждой последующей резервной копии журнала транзакций *в том порядке, в каком они были созданы*.

- При каждом добавлении или удалении межбазового ограничения ссылочной целостности или удалении таблицы, содержащей межбазовое ограничение, необходимо выполнить резервное копирование *обеих* баз данных, участвующих в ограничении.

  Предупреждение. Загрузка более ранних резервных копий этих баз данных может вызвать повреждение базы данных.

- Нельзя выполнять резервное копирование из сервера 11.x Adaptive Server в сервер 10.x Backup Server.
- Нельзя размещать резервные копии базы данных Sybase и данные, не относящиеся к Sybase (например, архивов UNIX), на одном ленточном устройстве.
- Нельзя выполнять команду dump transaction с параметрами with no_log или with truncate_only, если в базе данных есть автономные страницы.

Копирование журнала после сбоя устройства

- После сбоя устройства выполните команду dump transaction with no_truncate для копирования журнала без его очистки. Этот параметр можно использовать только в том случае, если журнал находится на отдельном сегменте, а база данных master доступна.
- Резервная копия, созданная командой dump transaction with no_truncate, является самой последней резервной копией журнала. При восстановлении базы данных эту резервную копию нужно загружать в последнюю очередь.

Резервное копирование баз данных, не имеющих отдельных журнальных сегментов

- Если журнальный сегмент базы данных находится на том же устройстве, что и сегменты данных, выполните команду dump transaction with truncate_only для удаления из журнала зафиксированных транзакций без создания резервной копии.

  Предупреждение. Команда dump transaction with truncate_only не позволяет провести восстановление базы данных. Чтобы базу данных можно было восстановить, необходимо при первой возможности выполнить команду dump database.

- Параметр with truncate_only нужно указывать для резервного копирования журналов транзакций баз данных master, model и sybsystemprocs, журнальные сегменты которых хранятся на том же устройстве, что и сегменты данных.

- Этот параметр можно также использовать для очень маленьких баз данных, в которых журнал транзакций и данные хранятся на одном устройстве.
- Для очень важных пользовательских баз данных журнальные сегменты не должны храниться на том же устройстве, что сегменты данных. Чтобы создать базу данных с отдельным журнальным сегментом, укажите инструкцию log on в команде create database или выполните команду alter database и системную процедуру sp_logdevice для переноса журнала на отдельное устройство.

Резервное копирование только завершенных транзакций

- Параметр with standby_access используется для резервного копирования журналов транзакций и загрузки журналов на сервер, который играет роль “теплого” резервного сервера базы данных.
- Если в команде dump transaction указан параметр with standby_access, то будут зарезервированы только записи, относящиеся к транзакциям, завершенным до начала открытых транзакций.
- Параметр dump tran[saction]...with standby_access необходимо указать, если два или более журналов транзакций будут загружаться последовательно, а база данных между загрузками должна оставаться в оперативном режиме.
- После загрузки резервной копии, которая была создана с помощью параметра with standby_access, нужно выполнить команду online database с параметром for standby_access, чтобы сделать базу данных доступной.

Предупреждение. Если журнал транзакций содержит открытые транзакции, а резервное копирование журнала выполняется без параметра with standby_access, то в версии 11.9.2 нельзя будет загрузить полученную таким образом резервную копию, перевести базу данных в оперативный режим, а затем загрузить следующую резервную копию журнала транзакций. Если загружается последовательность резервных копий транзакций, то базу данных можно перевести в оперативный режим только после загрузки резервной копии, созданной с параметром with standby_access, или после загрузки всей последовательности резервных копий.

Резервное копирование без журнала

Предупреждение. Команда dump transaction with no_log используется только как крайнее средство, когда из-за недостаточного свободного пространства в журнале нельзя применить обычный метод резервного копирования журнала транзакций (dump transaction или dump transaction with truncate_only). Команда dump transaction with no_log не позволяет впоследствии восстановить базы данных. Чтобы базу данных можно было восстановить, необходимо при первой возможности выполнить команду dump database.

- Команда dump transaction...with no_log очищает журнал без записи в журнал этой операции. Поскольку эта команда не копирует данные, то для ее выполнения требуется лишь имя базы данных.
- Каждое выполнение команды dump transaction...with no_log считается ошибкой и записывается в журнал ошибок сервера Adaptive Server.
- Этот параметр, скорее всего, не потребуется использовать, если журнальные сегменты базы данных хранятся на отдельном от сегментов данных устройстве, процедура последнего порога выполняет резервное копирование журнала транзакций достаточно часто, а для журнала и базы данных выделено достаточно пространства. Если возникла необходимость в использовании параметра with no_log, то резервное копирование должно выполняться чаще и пространства для журнала должно выделяться больше.

Планирование резервного копирования

- Резервное копирование журнала транзакций является *динамическим*: оно может выполняться, когда база данных активна. Это может несколько замедлить работу системы, поэтому резервное копирование должно выполняться в то время, когда обновления в базе данных происходят не слишком интенсивно.
- После создания базы данных нужно сразу же сделать копию всей базы данных с помощью команды dump database. В новой базе данных нельзя выполнять команду dump transaction до выполнения команды dump database.
- Должен быть разработан график регулярного резервного копирования пользовательских баз данных и их журналов транзакций.
- Команда dump transaction использует меньший объем памяти и требует меньше времени по сравнению с командой dump database. Обычно резервные копии журнала транзакций создаются чаще, чем резервные копии базы данных.

Использование порогов для автоматизации выполнения команды *dump transaction*

- Использование порогов позволяет автоматизировать процедуры резервного копирования. Чтобы можно было использовать последний порог в сервере Adaptive Server, журнальные сегменты пользовательских баз данных не должны храниться на том же устройстве, что сегменты данных.
- Когда пространство в журнальном сегменте становится меньше последнего порога, то сервер Adaptive Server выполняет процедуру последнего порога. Чтобы защитить систему от неблагоприятных последствий нехватки пространства для журнала, в процедуру последнего порога нужно включить команду dump transaction. Дополнительную информацию см. в описании процедуры sp_thresholdaction.
- С помощью процедуры sp_addthreshold можно добавить второй порог для контроля за пространством журнала. Дополнительную информацию о порогах см. в книге *Руководство по системному администрированию*.

Указание устройств для хранения резервных копий

- Устройство для хранения резервных копий можно задать с помощью литерала, локальной переменной или параметра хранимой процедуры.
- Можно указать локальное устройство для хранения резервных копий следующим образом:
  - как имя логического устройства из системной таблицы sysdevices;
  - указав полный путь;
  - указав относительный путь.

  Сервер Backup Server разрешает относительные имена, используя текущий рабочий каталог сервера Adaptive Server.
- При резервном копировании на ленточные и дисковые устройства можно расщеплять резервную копию. Помещать несколько резервных копий на одно устройство можно только при резервном копировании на ленточные устройства.
- При резервном копировании по сети нужно указать полный путь для устройства для хранения резервных копий. Путь должен быть допустимым для машины, на которой работает сервер Backup Server. Если имя включает какие-либо символы помимо букв, цифр или символа подчеркивания (_), его необходимо заключить в кавычки.

- При выполнении команд dump могут возникнуть проблемы, связанные с владением и полномочиями на устройство для хранения резервных копий. Системная процедура sp_addumpdevice добавляет устройство в системную таблицу, но не гарантирует того, что можно будет выполнить резервное копирование на это устройство или создать файл как устройство для хранения резервных копий.
- Можно выполнять несколько операций резервного копирования (или загрузок) одновременно, но каждая операция должна работать со своим устройством для хранения резервных копий.

Определение характеристик ленточного устройства

- Если в команде dump transaction не указан параметр init и сервер Backup Server не может определить тип устройства, то эта команда не будет выполнена. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

Серверы Backup Server

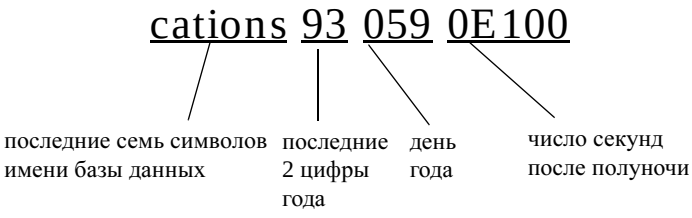
- Сервер Backup Server должен работать на той же машине, что и сервер Adaptive Server. Запись о сервере Backup Server должна содержаться в таблице master..sysservers. Эта запись создается в процессе установки системы или перехода на новую версию, поэтому ее не следует удалять.
- Если устройства для хранения резервных копий размещены на другой машине и резервное копирование выполняется через сеть, то необходимо, чтобы на удаленной машине был установлен сервер Backup Server.

Файлы резервной копии

- Резервное копирование журнала с параметром init перезаписывает все существующие файлы на ленте или диске.
- Из имен файлов резервной копии можно узнать базу данных, чьей резервной копией являются эти файлы, и время выполнения резервного копирования. Если имя файла не указывается, то сервер Backup Server создаст файл с именем, сформированным путем конкатенации следующих элементов:
  - Последних семи символов имени базы данных.
  - Двух цифр, обозначающих год.
  - Трех цифр, обозначающих день года (от 1 до 366).
  - Времени создания файла резервной копии в шестнадцатеричном формате.

Например, файл *cations930590E100* содержит копию базы данных publications, созданную в пятьдесят девятый день 1993 года:

Рис. 7-4. Правило именования резервных копий журнала транзакций



- Сервер Backup Server отправляет имя файла резервной копии получателю, указанному в предложении with notify. К ленточному устройству с резервной копией следует прикрепить метку, содержащую имя базы данных, имя файла, дату резервного копирования и другую важную информацию. Если лента, с которой загружается база данных, не имеет такой опознавательной метки, нужно указать параметры with headeronly и with listonly, чтобы можно было определить содержимое ленты.

Имена томов

- Метки томов резервных копий соответствуют стандарту ANSI по меткам для лент. Метка содержит номер логического тома и позицию устройства в наборе резервных копий.
- В процессе загрузки сервер Backup Server использует метку ленты для проверки того, что тома смонтированы в правильном порядке. Это позволяет загружать данные с меньшего числа устройств по сравнению с тем, которое было использовано при резервном копировании.

Примечание. Когда резервное копирование или загрузка происходит через сеть, для всех операций необходимо указать одно и то же число устройств для хранения резервных копий.

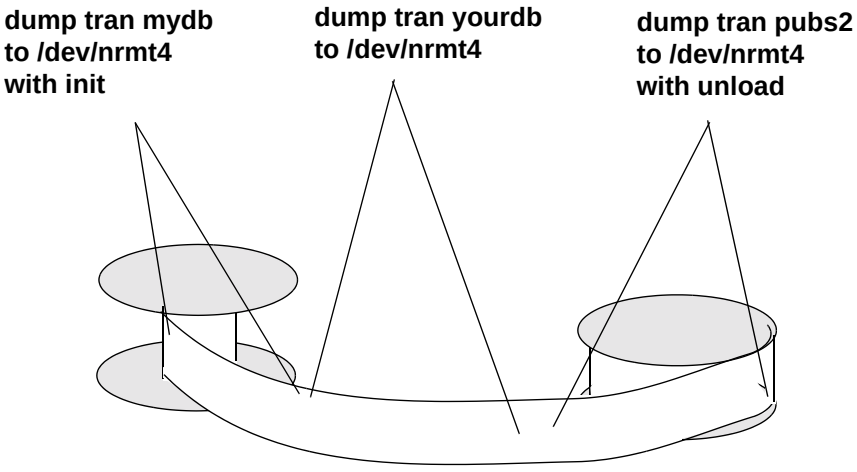
Изменение томов резервной копии

- *В операционной системе UNIX* сервер Backup Server просит сменить том при заполнении ленточного устройства. После того как смонтирован другой том, нужно уведомить об этом сервер Backup Server, выполнив системную процедуру sp_volchanged на любом сервере Adaptive Server, взаимодействующем с сервером Backup Server.
- Если сервер Backup Server обнаружил сбой в текущем смонтированном томе (например, если смонтирован неверный том), то он посылает клиенту или консоли оператора сообщение о смене тома. При получении этого сообщения нужно выполнить системную процедуру sp_volchanged.

Добавление к тому и перезапись тома

- По умолчанию сервер Backup Server использует параметр noinit, то есть записывает последовательные резервные копии на один и тот же том ленты, позволяя эффективно использовать ленточные носители с большим объемом пространства. Данные добавляются за последним маркером окончания ленты. Новые резервные копии можно добавлять только к последнему тому многотомной резервной копии. Прежде чем делать запись на ленту, сервер Backup Server проверяет, не истек ли срок действия первого файла. Если лента содержит данные, которые не относятся к Sybase, то сервер не производит на нее запись, чтобы не удалять потенциально ценную информацию.
- Параметр init используется для повторной инициализации тома. Если указывается параметр init, сервер Backup Server перезаписывает любую существующую информацию, даже если лента содержит данные, не относящиеся к Sybase, или срок первого файла еще не истек, или лента имеет ограничения доступа ANSI.
- На Рис. 7-5 показано резервное копирование трех журналов транзакций на один том. Можно использовать следующие параметры:
  - init – для инициализации ленты для первой резервной копии;
  - noinit (по умолчанию) – для добавления последовательных резервных копий;
  - unload – для перемотки и выгрузки ленты после последней операции резервного копирования.

Рис. 7-5. Резервное копирование трех журналов транзакций на один том



Резервное копирование журналов, хранящихся на зеркальном наборе устройств

- В начале выполнения команды dump transaction сервер Adaptive Server передает серверу Backup Server имена всех первичных устройств, на которых хранятся журналы. Если для первичного устройства отключен режим зеркального копирования, сервер Adaptive Server передает вместо него имя вторичного устройства. Если до того, как сервер Backup Server завершил передачу данных, происходит сбой какого-либо именованного устройства, то сервер Adaptive Server аварийно завершает резервное копирование.
- Если во время выполнения команды dump transaction попытаться отключить режим зеркального копирования для именованного устройства для хранения журнала, то сервер Adaptive Server выдаст соответствующее сообщение. Пользователь, выполняющий команду disk unmirror, может завершить резервное копирование или отложить выполнение команды disk unmirror до завершения резервного копирования.
- Команды dump transaction with truncate_only и dump transaction with no_log не используют сервер Backup Server. Эти команды можно выполнять, даже если для устройства для хранения журнала отключен режим зеркального копирования (в результате сбоя устройства или выполнения команды disk unmirror).

- Команда dump transaction копирует только сегмент журнала. Она будет работать, даже если для устройства, предназначенного только для данных, будет отключен режим зеркального копирования (в результате сбоя устройства или выполнения команды disk unmirror).

Стандарты

Соответствие стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду dump transaction могут выполнять только системные администраторы, пользователи с ролью Operator и владелец базы данных.

См. также

Команды dump database, load database, load transaction, online database

Системные процедуры sp_addumpdevice, sp_dboption, sp_dropdevice, sp_helpdevice, sp_logdevice, sp_volchanged

execute

Описание

Запускает процедуру или динамически выполняет команды Transact-SQL.

Синтаксис

```
[exec[ute]] [@return_status = ]
        [[[сервер .]база_данных.]владелец.]имя_процедуры[;число]
            [[@имя_параметра =] значение |
                [@имя_параметра =] @переменная []
            [,[@имя_параметра =] значение |
                [@имя_параметра =] @переменная []...]]
        [with recompile]
```

или

```
exec[ute] ("строка" | символьная_переменная
    [+ "строка" | символьная_переменная]...)
```

Параметры

execute | exec

Используется для выполнения хранимой процедуры или расширенной хранимой процедуры (extended stored procedure, ESP). Это ключевое слово обязательно, только если вызов хранимой процедуры – не первый оператор в пакете.

@return_status

Необязательная целочисленная переменная, которая хранит возвращаемое состояние хранимой процедуры. Переменная *@return_status* должна быть объявлена в пакете или хранимой процедуре до использования в операторе выполнения.

сервер

Имя удаленного сервера. Процедуру можно выполнять на другом сервере Adaptive Server, если есть полномочие на использование этого сервера и выполнение этой процедуры в соответствующей базе данных. Если указано имя сервера, но не указано имя базы данных, Adaptive Server ищет процедуру в базе данных по умолчанию.

база_данных

Имя базы данных. Его нужно указать, если процедура находится в другой базе данных. По умолчанию *база_данных* – это текущая база данных. Пользователь может выполнять процедуру в другой базе данных, если он является владельцем этой базы данных или имеет полномочие выполнять процедуру в этой базе данных.

владелец

Имя владельца процедуры. Его нужно указать, если в базе данных несколько процедур с этим именем. По умолчанию *владелец* – текущий пользователь. Пользователь, выполняющий эту процедуру, может не указывать владельца, только если процедура принадлежит ему или владельцу базы данных.

имя_процедуры
Имя процедуры, определенной с помощью create procedure.

;число
Необязательное целое число, используемое для группировки процедур с одним и тем же именем, чтобы можно было удалить их вместе одним оператором drop procedure. Часто группируются процедуры, используемые в одном приложении. Например, если процедуры, работающие с приложением orders, называются *orderproc;1*, *orderproc;2* и т. д., следующий оператор удалит всю группу:

```
drop proc orderproc
```

После группировки процедур нельзя удалять отдельные процедуры группы. Например, нельзя выполнить оператор:

```
drop procedure orderproc;2
```

имя_параметра
Имя аргумента процедуры согласно определению этой процедуры в команде create procedure. Перед именем параметра должен стоять знак @.

Если используется форма "@имя_параметра = значение", порядок имен параметров и констант может отличаться от порядка в команде create procedure. Однако, если эта форма используется для какого-то одного параметра, она должна использоваться и для всех идущих за ним параметров.

значение
Значение параметра или аргумента процедуры. Если не используется форма "@имя_параметра = значение", значения параметров должны быть указаны в порядке, определенном в команде create procedure.

@переменная
Имя переменной, в которой будет сохранен возвращаемый параметр.

output
Указывает, что хранимая процедура должна вернуть выходной параметр. Соответствующий параметр в хранимой процедуре также должен быть объявлен с ключевым словом output.

Ключевое слово output может быть сокращено до out.

with recompile
Заставляет Adaptive Server скомпилировать новый план. Этот параметр нужно указать, если вводится нетипичное значение или данные значительно изменились. Измененный план используется при последующих выполнениях. Этот параметр игнорируется при выполнении расширенной системной процедуры (ESP).

строка
Текстовая строка, содержащая часть выполняемой команды Transact-SQL. Количество символов в этой строке не ограничено.

символьная_переменная
Имя переменной, содержащей текст команды Transact-SQL.

Примеры

Пример 1. Все три примера выполняют процедуру showind с параметром titles:

```
execute showind titles
exec showind @tabname = titles
```

Если это единственный оператор в пакете или файле, то допустим следующий синтаксис:

```
showind titles
```

Пример 2. Выполнение процедуры checkcontract на удаленном сервере GATEWAY и сохранение возвращаемого состояния, обозначающего удачное завершение или сбой, в переменной *@retstat*:

```
declare @retstat int
execute @retstat = GATEWAY.pubs.dbo.checkcontract
"409-56-4008"
```

Пример 3. Выполнение процедуры roy_check, которой передается три параметра. Третий параметр *@pc* – это параметр output. После выполнения процедуры возвращаемое значение будет находиться в переменной *@percent*:

```
declare @percent int
select @percent = 10
execute roy_check "BU1032", 1050, @pc = @percent
output
select Percent = @percent
```

Пример 4. Эта процедура отображает информацию о системных таблицах, если не задан параметр:

```
create procedure
showsysind @table varchar(30) = "sys%"
as
```

```
select sysobjects.name, sysindexes.name, indid
from sysindexes, sysobjects
where sysobjects.name like @table
and sysobjects.id = sysindexes.id
```

Пример 5. Выполнение процедуры xp_echo, которой передается значение “Hello World!”. Возвращаемое значение расширенной хранимой процедуры сохраняется в переменой *result*:

```
declare @input varchar(12), @in varchar(12),
    @out varchar(255), @result varchar(255)
select @input="Hello World!"
execute xp_echo @in = @input, @out= @result output
```

Пример 6. Последняя команда execute формирует команду Transact-SQL путем конкатенации строковых значений и символьных переменных и выполняет эту команду:

```
select name from sysobjects where id=3

declare @tablename char(20)
declare @columname char(20)
select @tablename="sysobjects"
select @columname="name"
execute ('select ' + @columname + ' from ' +
@tablename + ' where id=3')
```

Пример 7. Выполнение процедуры sp_who:

```
declare @sproc varchar(255)
select @sproc = "sp_who"
execute @sproc
```

Использование

- Результаты процедуры могут зависеть от базы данных, в которой выполняется процедура. Например, пользовательская системная процедура *sp_foo*, которая выполняет системную функцию db_name(), возвращает имя базы данных, из которой она запущена. При запуске из базы данных pubs2 она вернет значение “pubs2”:

```
exec pubs2..sp_foo
------------------------------
pubs2
(1 row affected, return status = 0)
```

  При запуске из базы данных sybsystemprocs она вернет значение “sybsystemprocs”:

```
exec sybsystemprocs..sp_foo
----------------------------
sybsystemprocs
(1 row affected, return status = 0)
```

- Существует два способа указания параметров – по позиции или с помощью следующего синтаксиса:

  @имя_параметра = *значение* |

  Если используется второй способ, не обязательно задавать параметры в порядке, определенном в create procedure.

  Если возвращенное процедурой значение параметра, для которого указано ключевое слово output, будет использоваться в других операторах пакета или процедуры, содержащей команду execute, то значение этого параметра должно передаваться переменной. Например:

  имя_параметра = *@имя_переменной*

  Во время выполнения расширенной хранимой процедуры нужно передавать либо все параметры по имени, либо все параметры по значению. Нельзя передавать часть параметров по значению, а часть – по имени в команде execute, вызывающей расширенную хранимую процедуру.

- Синтаксис Dynamic SQL – exec (@имя_параметра) – тоже приемлем, однако в этом случае запись будет длиннее. Например, команда динамического SQL exec (@sproc ="7") передает процедуре целочисленное значение 7, но то же самое можно записать короче: exec @sproc 7.

- Нельзя использовать столбцы типа text и image в качестве параметров хранимых процедур или значений, передаваемых параметрам.

- Выполнение процедуры, где ключевое слово output задано для параметра, который не был определен как возвращаемый в команде create procedure, вызывает ошибку.

- Если для параметра указано ключевое слово output, то справа от знака равенства для этого параметра нельзя указывать константу – значение возвращаемого параметра должно передаваться переменной. Нужно объявить тип данных переменной и перед выполнением процедуры присвоить этой переменной значение. Возвращаемые параметры не могут иметь тип данных text или image.

- Ключевое слово execute можно не использовать, если оператор является первым в пакете. Пакет – это сегмент входного файла, оканчивающийся словом "go" на отдельной строке.
- Поскольку план выполнения для процедуры сохраняется во время ее первого выполнения, впоследствии она выполняется намного быстрее, чем эквивалентный набор отдельных операторов.
- Вложением называется ситуация, когда одна хранимая процедура вызывает другую. Уровень вложенности увеличивается при вызове процедуры и уменьшается при ее завершении. Если уровень вложенности превышает 16, транзакция прекращается с ошибкой. Текущий уровень вложенности хранится в глобальной переменной *@@nestlevel*.
- Возвращаемые значения 0 и от -1 до -14 используются в настоящее время Adaptive Server для обозначения состояния выполнения хранимых процедур. Значения от -15 до -99 зарезервированы для будущего использования. См. список значений в описании команды return.
- Параметры не являются частью транзакций, поэтому если параметр был изменен в транзакции, для которой позже был выполнен откат, его значение не станет прежним. Значение, возвращаемое вызывающей программе, всегда является значением на момент выхода из процедуры.
- Если процедура содержит оператор select *, она не будет воспринимать столбцы, добавленные в таблицу после создания процедуры (даже если в команде create procedure был указан параметр with recompile). В этом случае необходимо удалить процедуру и создать ее заново.
- Откат команд, выполненных посредством вызовов удаленных процедур, невозможен.
- Параметр with recompile игнорируется при выполнении расширенной хранимой процедуры.

Динамическое выполнение команд Transact-SQL

- Команда execute может конкатенировать предоставленные строки и символьные переменные и выполнять сформированную команду Transact-SQL. Форма команды execute может использоваться в пакетах SQL, процедурах и триггерах.

- Нельзя формировать с помощью строк и символьных переменных команды: begin transaction, commit, connect to, declare cursor, rollback, dump transaction, dbcc, set, use и вложенные команды execute.
- В команде execute может быть указана команда create view, но только если команда execute входит в пакет SQL. Если команда execute входит в процедуру, то в ней нельзя указывать create view ни как статическую команду, ни как строковый параметр.
- Значения параметров *строка* или *символьная_переменная* не могут содержать локальные переменные, объявленные в пакете SQL или процедуре.
- Допустимо указывать такие значения параметров *строка* и *символьная_переменная*, конкатенация которых образует команду, создающую таблицы. Однако внутри одного пакета SQL или процедуры таблица, созданная с помощью команды execute(), видна только другим командам execute(). После завершения пакета SQL или процедуры динамически созданная таблица становится постоянной и видимой другим командам.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию полномочия на выполнение команды execute принадлежат владельцу процедуры, который может передать их другим пользователям.

Полномочие на выполнение команд Transact-SQL, определенных с помощью параметров *строка* или *символьная_переменная*, проверяется по имени пользователя, выполняющего команду. Это справедливо, даже если команда execute() определена внутри процедуры или триггера, принадлежащего другому пользователю.

См. также

Команды create procedure, drop procedure, return

Системные процедуры sp_addextendedproc, sp_depends, sp_dropextendedproc, sp_helptext

fetch

Описание

Возвращает строку или набор строк из результирующего набора курсора.

Синтаксис

fetch *имя_курсора* [into *список_принимающих_переменных*]

Параметры

имя_курсора
Имя курсора.

into список_принимающих_переменных
Разделенный запятыми список параметров или локальных переменных, куда помещаются результаты курсора. Параметры и переменные должны быть объявлены до вызова команды fetch.

Примеры

Пример 1. Возвращение строки из результирующего набора курсора authors_crsr:

```
fetch authors_crsr
```

Пример 2. Возвращение строки из результирующего набора курсора, определенного курсором pubs_crsr в переменные *@name*, *@city* и *@state*:

```
fetch pubs_crsr into @name, @city, @state
```

Использование

Ограничения

- Прежде чем выполнять команду fetch, нужно объявить курсор и открыть его командой open.
- Значение *имя_курсора* не может быть параметром Transact-SQL или локальной переменной.
- С помощью команды fetch нельзя возвратить одну и ту же строку более одного раза. Нельзя идти назад по результирующему набору, однако можно закрыть и заново открыть курсор, чтобы снова создать результирующий набор и читать его с начала.
- Необходимо отношение “один к одному” между переменными в *списке_принимающих_переменных* и выражениями в списке выборки *оператора_select*, определяющего курсор. Типы данных переменных или параметров должны быть совместимы с типами данных столбцов в результирующем наборе курсора.
- Если включен режим связанных транзакций, Adaptive Server неявно начинает транзакции с оператора fetch, если в этот момент нет активных транзакций. Однако такая ситуация возникает только в случае, если параметр close on endtran установлен в *on* и курсор остается открытым после окончания открывшей его транзакции (поскольку оператор open также автоматически начинает транзакцию).

Позиция курсора

- Когда командой fetch выбраны все строки, курсор указывает на последнюю строку результирующего набора. Если снова выполнить команду fetch, через переменную *@@sqlstatus*, будет возвращено предупреждение о том, что больше нет данных, и позиция курсора сместится за границу набора результатов. Когда курсор находится в этой позиции, через него нельзя обновлять и удалять строки.
- Если при выполнении команды fetch into возникает ошибка, связанная с несоответствием количества переменных в *списке_принимающих_переменных* и количества выражений в запросе, определяющем курсор, позиция курсора не перемещается вперед. Однако в случае ошибки, связанной с несовместимостью типов данных переменных и столбцов в результирующем наборе курсора, позиция курсора продвигается.

Определение количества выбранных строк

- Команда fetch может одновременно выбрать несколько строк. Для этого нужно задать количество выбираемых строк с помощью команды set cursor rows.

Получение информации о выборках

- Глобальная переменная *@@sqlstatus* содержит информацию о состоянии, полученную при выполнении команды fetch (исключения-предупреждения). Допустимые значения *@@sqlstatus*: 0, 1 или 2 (см. таблицу 7-24).

Таблица 7-24. Значения переменной @@sqlstatus

0	Оператор fetch выполнен успешно.
1	Ошибка при выполнении оператора fetch.
2	В результирующем наборе больше нет данных. Это предупреждение может появиться, если текущая позиция курсора находится в последней строке результирующего набора и клиент выполняет оператор fetch для этого курсора.

Только оператор fetch может присвоить значение переменной *@@sqlstatus*. Другие операторы не влияют на ее значение.

- Глобальная переменная *@@rowcount* содержит количество строк, возвращенных клиенту из результирующего набора курсора, вплоть до последнего вызова fetch. Другими словами, она представляет суммарное количество строк, видимых клиенту в данный момент времени.

После того как все строки из результирующего набора курсора были прочитаны, *@@rowcount* представляет суммарное количество строк в результирующем наборе курсора. Каждый открытый курсор связан с определенной переменной *@@rowcount*, которая удаляется при закрытии курсора. Проверьте *@@rowcount* после выполнения команды fetch, чтобы выяснить количество считанных строк из курсора, указанного в этой команде fetch.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Использование переменных в целевом списке и выборка нескольких строк – расширения Transact-SQL.

Полномочия

По умолчанию команду fetch могут выполнять все пользователи.

См. также

Команды declare cursor, open, set

goto label

Описание

Передает управление метке, определенной пользователем.

Синтаксис

метка:
goto *метка*

Примеры

Здесь показано использование метки с именем restart:

```
declare @count smallint
select @count = 1
restart:
  print "yes"
select @count = @count + 1
while @count <=4
  goto restart
```

Использование

- Имя метки должно соответствовать правилам для идентификаторов, и при объявлении за ним должно следовать двоеточие (:). Двоеточие не ставится, если имя метки указывается с goto.
- Сделайте оператор goto зависимым от условия if, while или другого условия, чтобы предотвратить бесконечный цикл между goto и меткой.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

По умолчанию команду goto могут выполнять все пользователи. Для выполнения этой команды никакие полномочия не требуются.

См. также

Команды if...else, while

grant

Описание

Назначает полномочия пользователям или пользовательским ролям. Назначает роли пользователям, системным или пользовательским ролям.

Синтаксис

Чтобы разрешить доступ к объектам базы данных:

```
grant {all [privileges]| список_полномочий}
    on { имя_таблицы [(список_столбцов)]
        | имя_представления[(список_столбцов)]
        | имя_хранимой_процедуры}
    to {public | список_имен | имя_роли}
    [with grant option]
```

Чтобы разрешить выполнение конкретной команды:

```
grant {all [privileges] | список_команд}
    to {public | список_имен | имя_роли}
```

Чтобы назначить роль пользователю или другой роли:

```
grant {role предоставляемая_роль [, предоставляемая_роль  ...]}
    to получатель_роли [, получатель_роли...]
```

Параметры

all

Если ключевое слово all используется для предоставления полномочий на доступ к объектам базы данных (первая форма синтаксиса), оно указывает, что будут предоставлены все полномочия, применимые к данному объекту. Все владельцы объектов могут предоставлять доступ к собственным объектам командой grant all.

Только системный администратор и владелец базы данных могут предоставлять полномочия на создание объектов базы данных (вторая форма синтаксиса). Если команда grant all выполняется системным администратором, она предоставляет все полномочия create (create database, create default, create procedure, create rule, create table и create view). Если команда grant all выполняется владельцем базы данных, Adaptive Server предоставляет все полномочия create, кроме create database, и выдает информационное сообщение.

Команда grant all не предоставляет полномочия на выполнение команд set proxy и set session authorization.

список_полномочий

Список предоставляемых полномочий на доступ к объекту. Если в списке больше одного полномочия, их нужно разделить запятыми. В таблице приведены полномочия для каждого типа объекта:

Объект	список_полномочий может содержать
Таблица	select, insert, delete, update, references
Представление	select, insert, delete, update
Столбец	select, update, references Имена столбцов могут быть указаны в *списке_полномочий* или в *списке_столбцов* (см. пример 2).
Хранимая процедура	execute

список_команд

Список команд, которые может выполнять пользователь. Если в списке больше одной команды, их нужно разделить запятыми. Список команд может включать create database, create default, create procedure, create rule, create table, create view, set proxy и set session authorization.

Полномочие на create database может быть выдано только системным администратором и только из базы данных master.

Только администратор по безопасности может предоставлять пользователям полномочия на выполнение команд set proxy или set session authorization. Предоставление полномочий на выполнение set proxy или set session authorization позволяет получателю работать на сервере под именем другого пользователя. Команды set proxy и set session authorization идентичны, за исключением того что set session authorization соответствует стандарту ANSI92, а set proxy является расширением Transact-SQL.

имя_таблицы

Имя таблицы, на которую предоставляются полномочия. Таблица должна быть в текущей базе данных. В операторе grant может быть указан только один объект.

список_столбцов

Список столбцов, разделенный запятыми, к которым применяются полномочия. Если столбцы указаны, могут быть предоставлены полномочия только на команды select, references и update.

имя_представления

Имя представления, на которое предоставляются полномочия. Представление должно быть в текущей базе данных. В операторе grant может быть указан только один объект.

имя_хранимой_процедуры
Имя хранимой процедуры, на которую предоставляются полномочия. Хранимая процедура должна быть в вашей текущей базе данных. В операторе grant может быть указан только один объект.

public
Обозначает всех пользователей. Применительно к полномочиям на доступ к объекту группа public не включает владельца этого объекта. Применительно к полномочиям на создание объектов или выполнение команды set proxy группа public не включает владельца базы данных. Группе public, а также другим группам и ролям нельзя предоставлять полномочия командой grant с параметром with grant option.

список_имен
Список имен пользователей базы данных и/или имен групп, разделенный запятыми.

with grant option
Позволяет пользователям, перечисленным в *списке_имен*, предоставлять другим пользователям полномочия на доступ к объектам. Команда grant, содержащая параметр with grant option, не может содержать ключевое слово public, а также предоставлять полномочия какой-либо группе или роли.

role
Предоставляет роль пользователю, а также системной или пользовательской роли.

предоставляемая_роль
Имя системной или пользовательской роли, которую ответственный за безопасность системы предоставляет пользователю или роли.

получатель
Имя системной или пользовательской роли или имя пользователя, которому предоставляется роль.

имя_роли
Имя системной или пользовательской роли, которой предоставляется полномочие.

Примеры

Пример 1. Предоставление пользователю Mary и группе sales полномочий на выполнение команд insert и delete с таблицей titles:

```
grant insert, delete
on titles
to mary, sales
```

Пример 2. Два способа предоставить полномочие на выполнение команды update для столбцов price и advance таблицы titles группе public (включающей всех пользователей):

```
grant update
on titles (price, advance)
to public
```

или:

```
grant update (price, advance)
on titles
to public
```

Пример 3. Предоставление пользователям Harry и Billy полномочий на выполнение команд set proxy и set session authorization для работы на сервере под именем другого пользователя:

```
grant set proxy to harry, billy
```

Пример 4. Предоставление пользователям с ролью sso_role полномочий на выполнение команд set proxy и set session authorization для работы на сервере под именем другого пользователя:

```
grant set session authorization to sso_role
```

Пример 5. Предоставление пользователям с ролью vip_role права работать на сервере под именем другого пользователя. Роль vip_role должна быть определена администратором по безопасности командой create role:

```
grant set proxy to vip_role
```

Пример 6. Предоставление пользователям Mary и John полномочий на выполнение команд create database и create table. Поскольку предоставляется полномочие на выполнение команды create database, эта команда grant может быть выполнена только системным администратором и только из базы данных master. Mary и John могут выполнять команду create table только в базе данных master:

```
grant create database, create table
to mary, john
```

Пример 7. Предоставление всем пользователям всех полномочий на таблицу titles:

```
grant all on titles
to public
```

Пример 8. Предоставление всем пользователям всех полномочий на создание объектов в текущий базе данных. Если эта команда выполняется системным администратором из базы данных master, она также предоставляет полномочия на выполнение команды create database:

```
grant all
to public
```

Пример 9. Предоставление пользователю Mary полномочий на выполнение команды update с таблицей authors, а также права передачи этого полномочия другим пользователям:

```
grant update on authors
to mary
with grant option
```

Пример 10. Предоставление пользователю Bob полномочий на выполнение команд select и update со столбцом price таблицы titles, а также права передачи этого полномочия другим пользователям:

```
grant select, update on titles(price)
to bob
with grant option
```

Пример 11. Предоставление администраторам по безопасности полномочий на выполнение хранимой процедуры new_sproc:

```
grant execute on new_sproc
to sso_role
```

Пример 12. Предоставление пользователю James полномочия на создание в других таблицах ограничений ссылочной целостности, ссылающихся на столбец price в таблице titles:

```
grant references on titles(price)
to james
```

Пример 13. Предоставление роли spesialist_role (со всеми ее полномочиями и привилегиями) роли doctor:

```
grant role specialist_role to doctor_role
```

Пример 14. Предоставление пользователю Mary роли doctor_role:

```
grant role doctor_role to mary
```

Использование

- В синтаксисе команды grant ключевое слово to можно заменить на ключевое слово from.
- Таблица 7-25 содержит полномочия по умолчанию для команд Transact-SQL на Adaptive Server. Пользователь под графой “Полномочия по умолчанию” – это пользователь самого нижнего уровня, которому автоматически предоставляется право выпол-

нять команду. Этот пользователь может предоставить (с помощью команды grant) или отозвать (с помощью команды revoke) полномочия, если оно допускает передачу. Пользователям более высоких уровней полномочия предоставляются автоматически, или (в случае владельцев баз данных) они могут получить соответствующее полномочие с помощью команды setuser.

Например, владелец базы данных не получает автоматически полномочий на доступ к объектам, принадлежащим другим пользователям. Владелец базы данных может получить такое полномочие, выдав себя за владельца объекта командой setuser, а затем выполнив оператор grant или revoke. Системным администраторам разрешен доступ ко всем командам и объектам в любое время.

Сценарий установки Adaptive Server назначает набор полномочий группе “public” (группе по умолчанию). Для этих разрешений не нужно выполнять операторы grant и revoke.

В таблице 7-25 нет администратора по безопасности системы, который не имеет особых полномочий для команд и объектов, но только для некоторых системных процедур.

Таблица 7-25. Полномочия на команды и объекты

Полномочие на оператор	**По умолчанию предоставлено**					**Может быть предоставлено/ отозвано**		
	Системный администратор	**Оператор**	**Владелец базы данных**	**Владелец объекта**	**Public**	**Нет**	**Нет**	**Неприменимо**
alter database			X			(1)		
alter role								X
alter table				X			X	
begin transaction					X			X
checkpoint			X				X	
commit					X			X
connect to						X		
create database	X					X		
create default			X			X		
create index				X			X	
create procedure			X			X		

Полномочие на оператор	По умолчанию предоставлено					Может быть предоставлено/отозвано		
	Системный администратор	Оператор	Владелец базы данных	Владелец объекта	Public	Нет	Нет	Неприменимо
create role								X
create rule			X			X		
create table			X		(2)	X (2)		
create trigger					X	X		
create view			X			X		
dbcc	Зависит от параметров. См. описание команды dbcc в этом руководстве.						X	
delete				X (3)		X		
disk init	X						X	
disk mirror	X							
disk refit	X							
disk reinit	X							
disk remirror	X							
disk unmirror	X						X	
drop с любым объектом				X			X	
dump database		X	X				X	
dump transaction		X	X				X	
execute				X (4)		X		
grant для объекта				X		X		
команда grant			X			X		
insert				X (3)		X		
kill	X						X	
load database		X	X				X	
load transaction		X	X				X	
print					X			X
raiserror					X			X
readtext				X		(5)		
revoke для объекта				X			X	
команда revoke			X				X	
rollback					X			X

Полномочие на оператор	По умолчанию предоставлено					Может быть предоставлено/отозвано		
	Системный администратор	Оператор	Владелец базы данных	Владелец объекта	Public	Нет	Нет	Неприменимо
save transaction					X			X
select				X (3)		X		
set					X			X
setuser			X				X	
shutdown	X						X	
truncate table				X			X	
update				X (3)		X		
update all statistics				X			X	
update partition statistics				X			X	
update statistics				X			X	
writetext				X		(6)		

(1) Передается вместе с правами владения базой данных
(2) Группа Public (все пользователи) вправе создавать временные таблицы, полномочий не требуется
(3) Если подразумевается представление, по умолчанию полномочия принадлежат владельцу представления
(4) По умолчанию разрешено владельцу хранимой процедуры
(5) Передается с полномочиями на команду select
(6) Передается с полномочиями на команду update
“Нет” значит, что использование команды никогда не ограничивается
“Неприменимо” значит, что использование команды всегда ограничено

- Пользователь может предоставлять полномочия на доступ только к объектам, находящимся в его текущей базе данных.
- Чтобы пользователь мог создать таблицу, содержащую ограничение ссылочной целостности, ссылающееся на таблицу другого пользователя, он должен иметь полномочие references на родительскую таблицу (см. пример 10). Родительская таблица также должна содержать ограничение unique или уникальный индекс по столбцам, на которые ссылается ограничение. Об ограничениях ссылочной целостности см. описание команды create table.
- Очередность выполнения команд grant и revoke имеет значение. В случае конфликта действует команда, выполненная последней.

- Пользователю может быть дано полномочие на доступ к представлению или хранимой процедуре, даже если он не имеет полномочий на объекты, на которые ссылается эта процедура или представление. Подробности см. в книге *Руководство по системному администрированию*.
- Все пользователи имеют полномочия на объявление курсоров, независимо от их полномочий на базовые таблицы или представления, указанные в операторе declare cursor. Курсоры не являются объектами Adaptive Server (в отличие, например, от таблиц), поэтому полномочия к ним неприменимы. Когда пользователь открывает курсор, Adaptive Server определяет наличие у пользователя полномочий select на объекты, определяющие набор результатов этого курсора. Эти полномочия проверяются при каждом открытии курсора.

  Если пользователю разрешен доступ к объектам, определенным курсором, Adaptive Server открывает курсор и позволяет пользователю командой fetch выбирать строки данных через курсор. Adaptive Server не проверяет полномочия при выполнении команд fetch. Впрочем, если пользователь выполняет команду delete или update через этот курсор, проводится обычная проверка полномочий на удаление или обновление данных из базовых объектов этого курсора.
- Оператор grant добавляет одну строку в системную таблицу sysprotects для каждого пользователя, группы или роли, получающих полномочия. Если затем с помощью команды revoke отозвать полномочия у пользователя или группы, эта строка будет удалена из таблицы sysprotects. Если полномочия отзываются только у некоторых членов группы, а не у всей группы, которой они были предоставлены, первоначальная строка сохранится и будет добавлена новая строка, соответствующая отзыву.
- Если пользователь наследует какое-либо полномочие, являясь членом группы, и то же полномочие дано ему явно, то строка не добавляется в таблицу sysprotects. Например, пусть группе public предоставляются полномочия select на столбец phone таблицы authors, а затем пользователю John, являющемуся членом группы public, предоставляются полномочия select на все столбцы таблицы authors. В этом случае в таблицу sysprotects добавляется строка, соответствующая предоставлению полномочий пользователю John. Эта строка содержит ссылки на все столбцы в таблице authors, кроме столбца phone, для которого у этого пользователя уже есть полномочие.

- По умолчанию команду create trigger разрешено выполнять всем пользователям. Когда у пользователя отзываются полномочия на создание триггеров, в таблицу sysprotects добавляется строка, соответствующая этому отзыву. Чтобы разрешить этому пользователю выполнять команду create trigger, нужно выполнить две команды grant. Первая команда удаляет строку, соответствующую отзыву, из таблицы sysprotects, а вторая вставляет строку о предоставлении полномочия. Если у пользователя были отозваны полномочия на создание триггеров, он не сможет создавать триггеры даже на таблицах, которыми владеет. Отзыв полномочий пользователя на создание триггеров действует только на базу данных, в которой выполняется соответствующая команда отзыва.
- Узнать о полномочиях можно с помощью следующих системных процедур:
  - sp_helprotect выдает информацию о полномочиях для объекта базы данных или для пользователя;
  - sp_column_privileges выдает информацию о полномочиях для одного или нескольких столбцов в таблице или представлении;
  - sp_column_privileges выдает информацию о полномочиях для всех столбцов в таблице или представлении;
  - sp_activeroles отображает все активные роли для текущего сеанса пользователя на Adaptive Server;
  - sp_displayroles отображает все роли, предоставленные другой роли, или полное дерево иерархии ролей в табличном формате.

Предоставление полномочий на создание объектов с помощью команды *grant all*

- Если в команде grant all указаны только имена пользователей или групп (но не указаны имена объектов), эта команда назначает следующие полномочия: create database, create default, create procedure, create rule, create table и create view. Полномочие create database может быть дано только системным администратором и только из базы данных master.
- Только владелец базы данных и системный администратор может выполнять команду grant all без указания имени объекта, чтобы предоставить полномочия на команды create пользователям или группам. Если владелец базы данных попытается выполнить команду grant all, будет выдано сообщение о том,

что только системный администратор может предоставлять полномочия create database. Все остальные полномочия, описанные выше, будут предоставлены.

- Все владельцы объектов могут выполнять команду grant all с именем объекта, чтобы предоставить полномочия на объекты, которыми они владеют. Команда grant all, в которой указано имя таблицы или представления, а также имена пользователей или групп, дает полномочия на команды delete, insert, select, и update для указанной таблицы.

Правила применения команды *grant with grant option*

- Нельзя предоставлять полномочия с параметром with grant option группе public или какой-то другой группе или роли.
- При предоставлении полномочий системный администратор считается владельцем объекта. Если системный администратор дает полномочие на объект другого пользователя, то в таблице sysprotects и в результатах процедуры sp_helprotect в качестве пользователя, предоставившего полномочия (столбец grantor), будет содержаться имя владельца.
- Информация по каждой команде grant хранится в системной таблице sysprotects со следующими исключениями.
  - Adaptive Server выдает информационное сообщение, если определенное полномочие предоставлено пользователю более одного раза одним и тем же пользователем. В этом случае сохраняются сведения только о первой команде grant.
  - Если две команды grant идентичны, за исключением того что в одной из них указан параметр with grant option, то будут сохранены сведения о команде grant with grant option.
  - Если две команды grant дают одинаковые полномочия на определенную таблицу какому-то пользователю и отличаются только тем, что в них указаны разные столбцы, эти команды будут считаться за одну. Например, следующие команды grant по своему действию эквивалентны:

```
grant select on titles(price, contract)
    to keiko
grant select on titles(advance) to keiko
grant select on titles(price, contract,
    advance)
to keiko
```

Предоставление полномочий на авторизацию прокси и авторизацию сеансов

- Полномочия на выполнение команд set proxy или set session authorization позволяют подключаться к Adaptive Server от имени другого пользователя. Команды set proxy и set session authorization идентичны с одним исключением: set session authorization соответствует стандарту SQL, а set proxy – расширение Transact-SQL.
- Чтобы предоставить полномочие set proxy или set session authorization, нужно быть администратором безопасности системы и находиться в базе данных master.
- Имя, которое указывается в команде grant set proxy, должно быть действительным именем пользователя в базе данных, то есть это имя должно быть в таблице sysusers в базе данных.
- Команда grant all *не* включает полномочия set proxy или set session authorization

Предоставление полномочий ролям

- Можно использовать команду grant для предоставления полномочий всем пользователям, которым была дана указанная роль. Эта роль может быть системной ролью, например sso_role или sa_role, или пользовательской ролью. Для пользовательской роли администратор безопасности системы должен создать роль командой create role.

  Однако полномочие grant execute не мешает индивидуально разрешить выполнение хранимой процедуры пользователям, у которых нет указанной роли. Например, чтобы убедиться, что только администратору безопасности системы может в любое время быть разрешено выполнять хранимую процедуру, задействуйте системную функцию proc_role в самой хранимой процедуре. Она проверяет, обладает ли вызывающий пользователь нужной ролью для выполнения процедуры. См. также описание proc_role.
- Полномочия, предоставленные ролям, приоритетнее полномочий, выданных пользователям или группам. Пусть пользователь John имеет роль System Security Officer (администратор безопасности), а роли sso_role было дано полномочие на доступ к таблице sales. Если у пользователя John отозвано индивидуальное полномочие на доступ к таблице sales, он по-прежнему может получать доступ к этой таблице, поскольку полномочие его роли приоритетнее индивидуальных полномочий.

Пользователи и группы пользователей

- Группы пользователей позволяют предоставлять и отзывать полномочия одним оператором для нескольких пользователей. Каждый пользователь может быть членом только одной группы и всегда является участником группы public.
- Владелец базы данных или системный администратор могут добавлять новых пользователей процедурой sp_adduser и создавать группы процедурой sp_addgroup. Чтобы позволить пользователям Adaptive Server работать с базой данных с ограниченными привилегиями, можно добавить пользователя-гостя процедурой sp_adduser и назначить ему ограниченные полномочия. Все пользователи с регистрационными именами могут получать доступ к базе данных в качестве гостя.
- Для удаления пользователя применяйте процедуру sp_dropuser. Для удаления группы применяйте процедуру sp_dropuser.

  Для добавления нового пользователя в любую группу, кроме public, служит процедура sp_adduser. Для смены группы пользователя служит процедура sp_changegroup.

  Для вывода списка участников группы выполните процедуру sp_helpgroup.
- Когда выполняется процедура sp_changegroup для изменения состава группы, она стирает кэш защиты в памяти командой:

  ```
  grant all to null
  ```

  чтобы кэш мог быть обновлен информацией из таблицы sysprotects. Чтобы напрямую модифицировать таблицу sysprotects, свяжитесь со службой технической поддержки Sybase.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Предоставление полномочий группам и предоставление полномочия set proxy – расширения Transact-SQL. Полномочие set session authorization (аналогично по своим функциям полномочию set proxy) соответствует стандарту ANSI.

Полномочия

Доступ к объектам базы данных. По умолчанию предоставлять полномочия на доступ к объектам базы данных посредством grant вправе владельцы объектов. Владелец объекта может предоставить другим пользователям полномочия на доступ к своим объектам базы данных.

Выполнение команд. Полномочие create database может быть предоставлено только системным администратором и только из базы данных master. Полномочие create trigger может быть предоставлено только администратором по безопасности системы.

Прокси-авторизация и авторизация сеансов. Только администратор по безопасности системы может предоставить полномочия на выполнение команд set proxy и set session authorization и только из базы данных master.

Роли. Предоставлять роли можно только из базы данных master. Только администратор безопасности системы может предоставить роли sso_role, oper_role или пользовательскую роль какому-либо пользователю или роли. Только системные администраторы могут предоставлять роль sa_role какому-либо пользователю или роли. Только пользователь, обладающий обеими ролями, – sa_role и sso_role – может предоставлять роль, содержащую sa_role.

См. также

Хранимые процедуры каталога sp_column_privileges

Команды revoke, setuser, set

Функции proc_role

Системные процедуры sp_addgroup, sp_adduser, sp_changedbowner, sp_changegroup, sp_dropgroup, sp_dropuser, sp_helpgroup, sp_helprotect, sp_helpuser, sp_role

group by и having

Описание

Эти инструкции используются в операторах select для деления таблицы на группы и возвращения только тех групп, которые соответствуют условиям в инструкции having.

Синтаксис

Начало оператора select

[group by [all] *выражение_без_агрегатов*
[, *выражение_без_агрегатов*]...]

[having *условия_поиска*]

Конец оператора select

Параметры

group by

Указывает группы, на которые будет поделена таблица, и, если в список выборки включены агрегатные функции, находит итоговое значение для каждой группы. Эти итоговые значения (каждое значение соответствует одной группе) отображаются в результатах в виде столбцов. На эти столбцы можно ссылаться в инструкции having.

В списке выборки перед инструкцией group by можно указывать агрегатные функции avg, count, max, min и sum (при этом выражением обычно является имя столбца). Подробности см. в разделе “Агрегатные функции” на стр. 50.

Таблица может быть сгруппирована по любой комбинации столбцов, то есть группы могут быть вложены друг в друга, как в примере 2.

all

Расширение Transact-SQL, включающее в результаты все группы, даже если ни один элемент группы не удовлетворяет условию where. Например:

```
select type, avg(price)
from titles
where advance > 7000
group by all type

type
----------------  ----------
UNDECIDED               NULL
business                2.99
mod_cook                2.99
popular_comp           20.00
psychology              NULL
trad_cook              14.99

(6 rows affected)
```

"NULL" в столбце, содержащем результаты агрегатной функции, указывает, что ни один элемент группы не удовлетворяет условию where. Действие инструкции having противоположно действию ключевого слова all.

выражение_без_агрегатов

Выражение, которое не включает в себя ни одного агрегата. Расширение Transact-SQL позволяет группировать по выражению, не содержащему агрегаты, так же, как по столбцу.

В инструкции group by нельзя указывать заголовок столбца или псевдонимы. Приведенный ниже синтаксис корректен:

```
select Price=avg(price), Pay=avg(advance),
Total=price * $1.15
from titles
group by price * $1.15
```

having

Задает условия для инструкции group by подобно тому, как where задает условия для инструкции select.

Условия поиска в инструкции having могут содержать агрегатные выражения. Во всех остальных отношениях инструкция having с условиями поиска идентична инструкции where. Ниже представлен пример инструкции having с агрегатами:

```
select pub_id, total = sum(total_sales)
from titles
where total_sales is not null
group by pub_id
having count(*)>5
```

Когда Adaptive Server оптимизирует запросы, он анализирует условия поиска в инструкциях where и having и определяет, какие условия являются аргументами поиска, которые могут использоваться для выбора наилучших индексов и плана запроса. Для отбора строк используются все условия поиска. Об аргументах поиска см. книгу *Руководство по настройке производительности*.

Примеры

Пример 1. Вычисление среднего кредита и суммы продаж для каждого типа книги:

```
select type, avg(advance), sum(total_sales)
from titles
group by type
```

Пример 2. Группировка результатов по столбцу type, а внутри каждой группы с одинаковым значением этого столбца – по столбцу pub_id:

```
select type, pub_id, avg(advance), sum(total_sales)
from titles
group by type, pub_id
```

Пример 3. Вычисление результатов для всех групп, но отображение только групп, в которых столбец type начинается с “p”:

```
select type, avg(price)
from titles
group by type
having type like 'p%'
```

Пример 4. Вычисление результатов для всех групп, но отображение только для групп, удовлетворяющих составному условию в инструкции having:

```
select pub_id, sum(advance), avg(price)
from titles
group by pub_id
having sum(advance) > $15000
and avg(price) < $10
and pub_id > "0700"
```

Пример 5. Вычисление суммы продаж для каждой группы (издателя) после соединения таблиц titles и publishers:

```
select p.pub_id, sum(t.total_sales)
from publishers p, titles t
where p.pub_id = t.pub_id
group by p.pub_id
```

Пример 6. Отображение заголовков книг, по которым аванс (столбец advance) превышает $1000, а цена (столбец price) – выше средней цены всех книг:

```
select title_id, advance, price
from titles
where advance > 1000
having price > avg(price)
```

Использование

- После ключевых слов group by может быть указано имя столбца или любое другое выражение (кроме заголовка столбца или псевдонима). Инструкцию group by также можно применять для вычисления результатов и для отображения столбца или выражения, которого нет в списке выборки (это расширение Transact-SQL описано в разделе “Расширения Transact-SQL для group by и having” на стр. 581).

- Максимальное количество столбцов или выражений, которое можно указать в инструкции group by, равно 31, что совпадает с максимальным числом индексов по таблице.
- Максимальный размер столбца и суммарный размер столбцов, указанных в инструкции group by, ограничены максимальным размером индекса для данного размера логической страницы. Дело в том, что Adaptive Server генерирует рабочую таблицу с ключом при группировке результатов. О размерах индексов см. раздел create index на стр. 358.

  Ограничение на размер индекса может вызвать ошибки при обработке инструкции group by. Например, инструкция group by в 1024-байтном столбце на сервере с размером страницы 2 КБ вызывает ошибку, если ограничение на размер столбца индекса – 600 байтов.
- Значения NULL в столбце group by помещаются в отдельную группу.
- В инструкциях group by и having нельзя указывать столбцы text или image.
- Нельзя использовать инструкцию group by в операторе select обновляемого курсора.
- Агрегатные функции можно использовать только в списке выборки и в инструкции having. Они недопустимы в инструкциях where и group by.

  Агрегатные функции бывают двух типов. Агрегатная функция, применяемая ко *всем строкам в таблице*, удовлетворяющим условию запроса (вычисляющая, таким образом, одно значение для всей таблицы), называется *скалярным агрегатом*. Агрегатная функция в списке выборки, не содержащем инструкции group by, применяется ко всей таблице; это один из вариантов скалярного агрегата.

  Агрегатная функция, применяемая к *группе строк в указанном столбце или выражении* (вычисляющая значение для каждой группы), называется *векторным агрегатом*. Для любого типа агрегата результаты агрегатных операций отображаются в новых столбцах, на которые может ссылаться инструкция having.

  Разрешается вкладывать векторный агрегат в скалярный. Подробности см. в разделе “Агрегатные функции.” на стр. 50.

Как работают запросы с инструкциями *group by* и *having* и агрегатами

- Инструкция where исключает строки, не удовлетворяющие ее условиям поиска; она одинаково работает для запросов с группировкой и обычных запросов.
- Инструкция group by собирает оставшиеся строки, формируя для каждого значения выражения, указанного в этой инструкции, свою группу. Если опустить group by, будет создана одна группа для всей таблицы.
- Агрегатные функции, указанные в списке выборки, вычисляют итоговые значения для каждой группы. Скалярные агрегаты вычисляют только одно значение для таблицы. Векторные агрегаты вычисляют значения для отдельных групп.
- Инструкция having исключает из результатов группы, которые не соответствуют условиям этой инструкции. Хотя инструкция having проверяет только строки, но, если присутствует инструкция group by, может показаться, что эта инструкция работает над группами.
    - Если в запросе содержится инструкция group by, то инструкция having исключает результирующие группы. Поэтому кажется, что having работает над группами.
    - Если в запросе нет инструкции group by, то инструкция having исключает результирующие строки из таблицы (состоящей из одной группы). Поэтому кажется, что инструкция having работает над строками (результаты аналогичны результатам применения инструкции where).

Стандартные запросы с инструкциями *group by* и *having*

- Все запросы с инструкциями group by и having в разделе “Примеры” соответствуют стандарту SQL. Согласно этому стандарту, запросы с инструкциями group by, having и векторными агрегатными функциями формируют одну строку и одно итоговое значение для каждой группы и должны соответствовать следующим правилам:
    - Столбцы, указанные в списке выборки, также должны быть перечислены в выражении group by или являться аргументами агрегатных функций.
    - Выражение в инструкции group by может содержать имена только тех столбцов, которые указаны в списке выборки. Однако оно не может содержать столбцов, указанных в списке выборки только в качестве аргументов агрегатных функций.

  - Столбцы в выражении having должны иметь одно значение (например, быть аргументами агрегатных функций) и должны присутствовать в списке выборки или в инструкции group by. Запросы с агрегатом в списке выборки и инструкцией having *должны* содержать инструкцию group by. Если для запроса без агрегата в списке выборки не указана инструкция group by, все строки, удовлетворяющие условию where, считаются одной группой (см. пример 6).

  В запросах без группировки принцип исключения строк инструкцией where кажется очевидным. В запросах с группировкой инструкция where исключает строки перед обработкой инструкции group by, а having исключает строки из окончательного результирующего набора.

- Стандарт SQL позволяет запросам, соединяющим две или более таблицы, содержать инструкции group by и having, если они соответствуют перечисленным выше условиям. При записи соединений или других сложных запросов используйте стандартный синтаксис инструкций group by и having, пока полностью не поймете действие этих инструкций в Transact-SQL (оно описано в разделе "Расширения Transact-SQL для group by и having").

  Чтобы избежать проблем с расширениями, в Adaptive Server предусмотрен параметр fipsflagger, устанавливаемый командой set, который выдает нефатальное предупреждение о каждом вхождении расширения Transact-SQL в запрос. Более подробную информацию см. в описании команды set.

Расширения Transact-SQL для *group by* и *having*

- Расширения стандартного SQL, появившиеся в Transact-SQL, позволяют более гибко отображать данные, разрешая указывать столбцы и выражения, которые не используются для создания групп и вычислений итоговых значений.

  - Список выборки, содержащий агрегаты, может также содержать столбцы, которые не являются аргументами агрегатных функций и не включены в инструкцию group by. Такие столбцы называются *расширенными*. Добавление расширенных столбцов приводит к появлению дополнительных строк в конечном наборе результатов.

  - Инструкция group by может включать столбцы и выражения, которые не содержатся в списке выборки.

- Инструкция group by all отображает все группы, даже те, что исключены из вычислений инструкцией where. См. пример для ключевого слова all в разделе “Параметры”.
- Инструкция having может содержать столбцы и выражения, которые не содержатся ни в списке выборки, ни в инструкции group by.

Если в результате использования расширений Transact-SQL в набор результатов попадают дополнительные строки и столбцы или если не указана инструкция group by, результаты запроса будет сложно интерпретировать. Следующие примеры помогут понять, как расширения Transact-SQL могут влиять на результаты запросов.

- Эти примеры иллюстрируют различие между запросами на основе стандартных инструкций group by и having и запросами на основе расширений Transact-SQL.

  a Пример запроса со стандартной группировкой:

```
select type, avg(price)
from titles
group by type

type
----------------------  ----------
UNDECIDED                 NULL
business                 13.73
mod_cook                 11.49
popular_comp             21.48
psychology               13.50
trad_cook                15.96

(6 rows affected)
```

  b В следующем примере в список выборки включен “расширенный” столбец price, которого нет ни в агрегате, ни в инструкции group by. Благодаря этому будут отображены все удовлетворяющие условиям запроса строки во всех группах (в которых есть такие строки), даже при том, что стандартная инструкция group by выдает только одну строку на группу. Тем не менее, векторный агрегат все же будет вычислен, так как инструкция group by определяет среднюю цену (значения столбца price) для каждой группы (это те же значения, что вычислялись для примера a). В каждой строке будет отображена средняя цена для группы, содержащей эту строку:

```
select type, price, avg(price)
from titles
group by type

type           price
------------ --------------- --------------
business       19.99            13.73
business       11.95            13.73
business        2.99            13.73
business       19.99            13.73
mod_cook       19.99            11.49
mod_cook        2.99            11.49
UNDECIDED       NULL             NULL
popular_comp   22.95            21.48
popular_comp   20.00            21.48
popular_comp    NULL            21.48
psychology     21.59            13.50
psychology     10.95            13.50
psychology      7.00            13.50
psychology     19.99            13.50
psychology      7.99            13.50
trad_cook      20.95            15.96
trad_cook      11.95            15.96
trad_cook      14.99            15.96

(18 rows affected)
```

- Обработка расширенных столбцов Transact-SQL выглядит так, будто запрос игнорирует инструкцию where. Этот запрос вычисляет средние цены на основе только тех строк, которые удовлетворяют инструкции where, но отображает и те строки, которые не удовлетворяют условию этой инструкции.

  Сначала Adaptive Server формирует рабочую таблицу, содержащую только значения столбца type и агрегатные значения, отбирая строки по условию инструкции where. Эта таблица соединяется с таблицей titles по столбцу группировки type, чтобы включить столбец price в результаты, однако условие инструкции where в этом соединении *не* проверяется.

  Единственная строка в таблице titles, которая не вошла в результат, – это строка со значением type, равным “UNDECIDED”, и ценой NULL, так как для этой строки не было результатов в рабочей таблице. Если также нужно исключить из отображаемых результатов строки, в которых цена меньше $10.00, добавьте инструкцию having, повторяющую инструкцию where, как показано в примере 4:

```
select type, price, avg(price)
from titles
where price > 10.00
group by type
```

```
type           price
------------ ---------------- --------------
business        19.99            17.31
business        11.95            17.31
business         2.99            17.31
business        19.99            17.31
mod_cook        19.99            19.99
mod_cook         2.99            19.99
popular_comp    22.95            21.48
popular_comp    20.00            21.48
popular_comp     NULL            21.48
psychology      21.59            17.51
psychology      10.95            17.51
psychology       7.00            17.51
psychology      19.99            17.51
psychology       7.99            17.51
trad_cook       20.95            15.96
trad_cook       11.95            15.96
trad_cook       14.99            15.96

(17 rows affected)
```

d Если в инструкции having указываются дополнительные условия, например агрегаты, не забудьте также включить в эту инструкцию все условия, указанные в инструкции where. В противном случае по набору результатов будет казаться, что условия, содержащиеся в инструкции where, но отсутствующие в инструкции having, игнорируются:

```
select type, price, avg(price)
from titles
where price > 10.00
group by type
having price > 10.00
```

```
type           price
---------- ----------------  --------------
business       19.99           17.31
business       11.95           17.31
business       19.99           17.31
mod_cook       19.99           19.99
popular_comp   22.95           21.48
popular_comp   20.00           21.48
```

```
psychology     21.59          17.51
psychology     10.95          17.51
psychology     19.99          17.51
trad_cook      20.95          15.96
trad_cook      11.95          15.96
trad_cook      14.99          15.96

(12 rows affected)
```

e Это пример стандартного запроса с группировкой на основе соединения двух таблиц. Он группирует по столбцу pub_id, а затем по столбцу type для каждого идентификатора издателя (столбец pub_id), чтобы вычислить векторный агрегат для каждой строки:

```
select p.pub_id, t.type, sum(t.total_sales)
from publishers p, titles t
where p.pub_id = t.pub_id
group by p.pub_id, t.type
```

```
pub_id  type
------  ------------  --------
0736    business       18722
0736    psychology      9564
0877    UNDECIDED       NULL
0877    mod_cook       24278
0877    psychology       375
0877    trad_cook      19566
1389    business       12066
1389    popular_comp   12875

(8 rows affected)
```

Может показаться, что результаты следующего примера, в котором в список выборки добавлен расширенный столбец pub_name, а инструкция group by оставлена без изменений, не будут принципиально отличаться от результатов предыдущего:

```
select p.pub_id, p.pub_name, t.type,
    sum(t.total_sales)
from publishers p, titles t
where p.pub_id = t.pub_id
group by p.pub_id, t.type
```

Однако результаты этого запроса будут сильно отличаться от результатов первого запроса в этом примере. После соединения двух таблиц для определения векторного агрегата в рабочей таблице Adaptive Server соединяет рабочую таблицу

с таблицей publishers, содержащей расширенный столбец для получения окончательных результатов. Количество соединений будет равно количеству таблиц, содержащих расширенные столбцы, участвующие в запросе.

Как можно видеть, использование расширенных столбцов в запросах, соединяющих таблицы, может дать непонятные результаты. В большинстве случаев следует использовать стандартную инструкцию group by в запросах с соединением таблиц.

f Этот пример использует расширение Transact-SQL для group by, позволяющее включать столбцы, которых нет в списке выборки. Группировка результатов для вычисления векторного агрегата производится по столбцам pub_id и type. Однако окончательные результаты не включают столбец type. В этом случае может потребоваться знать только количество разных типов книг, проданных каждым издателем (столбец pub_id):

```
select p.pub_id, sum(t.total_sales)
from publishers p, titles t
where p.pub_id = t.pub_id
group by p.pub_id, t.type
```

```
pub_id
------  --------
0736       18722
0736        9564
0877        NULL
0877        24278
0877          375
0877        19566
1389        12066
1389        12875

(8 rows affected)
```

g Этот пример объединяет два эффекта расширений Transact-SQL. Во-первых, он опускает инструкцию group by, хотя содержит агрегат в списке выборки. Во-вторых, он включает расширенный столбец. Пропуск инструкции group by дает следующее:

- Таблица будет единой группой. Скалярный агрегат определит, что три строки удовлетворяют запросу.

- Столбец pub_id становится расширенным столбцом Transact-SQL, поскольку он отсутствует в инструкции group by. Инструкции having нет, поэтому все строки группы будут отображены.

```
select pub_id, count(pub_id)
from publishers

pub_id
---------- ---------
0736               3
0877               3
1389               3

(3 rows affected)
```

h Инструкция where исключает издателей со значением pub_id выше 1000 из единой группы, поэтому скалярная агрегатная функция выдаст, что две строки удовлетворяют условию запроса. Благодаря тому, что в список выборки включен "расширенный" столбец pub_id, будут отображены все строки таблицы publishers, удовлетворяющие условию запроса:

```
select pub_id, count(pub_id)
from publishers
where pub_id < "1000"

pub_id
-------------- -----------
0736                     2
0877                     2
1389                     2

(3 rows affected)
```

i Этот пример иллюстрирует действие инструкции having, использованной без инструкции group by.

- Таблица считается единой группой. Так как инструкция where отсутствует, функция count подсчитывает количество всех строк в группе (таблице).
- Строки в таблице с одной группой проверяются на соответствие условию инструкции having.
- В результате будет выведено две строки, удовлетворяющие условию запроса.

```
select pub_id, count(pub_id)
from publishers
having pub_id < "1000"

pub_id
-------------- ---------
0736                   3
0877                   3
(2 rows affected)
```

j В этом примере используется расширенный синтаксис инструкции having, позволяющий указывать в ней столбцы и выражения, которых нет в списке выборки и инструкции group by. Здесь определяется средняя цена каждого типа книг, кроме типов, суммарные продажи которых не превышают $10 000, хотя в результатах агрегат sum не отображается:

```
select type, avg(price)
from titles
group by type
having sum(total_sales) > 10000

type
------------  ----------
business           13.73
mod_cook           11.49
popular_comp       21.48
trad_cook          15.96

(4 rows affected)
```

Инструкции *group by* и *having* и порядки сортировки

- Если порядок сортировки на сервере не зависит от регистра, инструкция group by игнорирует регистр группируемых столбцов. Например, если эти данные хранятся на сервере, не различающем регистр букв:

```
select lname, amount
from groupdemo
lname           amount
---------- ------------------
Smith               10.00
smith                5.00
SMITH                7.00
Levi                 9.00
Lйvi                  20.00
```

то группировка по столбцу lname дает такие результаты:

```
select lname, sum(amount)
from groupdemo

 lname
lname
---------- -----------------
Levi                 9.00
Lйvi                20.00
Smith                22.00
```

Тот же запрос на сервере, нечувствительном к регистру и диакритическим знакам, даст такие результаты:

```
 lname
 ---------- -----------------
 Levi                 29.00
 Smith                  22.00
```

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Использование столбцов внутри списка select, которых нет в списке group by и по которым нет агрегатных функций, – расширение Transact-SQL.

Использование ключевого слова all – расширение Transact-SQL.

См. также

Команды compute, declare, select, Инструкция where

Функции Агрегатные функции

if...else

Описание

Накладывает условия на выполнение оператора SQL.

Синтаксис

if *логическое_выражение* [plan "*абстрактный план*"]
операторы

[else
[if *логическое_выражение*] [plan "*абстрактный план*"]
оператор]

Параметры

логическое_выражение

Выражение (имя столбца, константа, любая комбинация имен столбцов и констант, соединенных знаками арифметических или поразрядных операторов, либо подзапрос), возвращающее значение TRUE, FALSE или NULL. Если в выражении содержится команда select, она должна быть заключена в круглые скобки.

команды

Одна команда SQL или блок команд, заключенный между командами begin и end.

plan "абстрактный план"

Определяет абстрактный план для оптимизации запроса. Это может быть полный или частичный план, заданный на языке абстрактных планов. В инструкциях if планы можно задавать только для тех выражений, которые поддаются оптимизации, то есть для запросов к таблицам. Дополнительную информацию см. в главе 30, “Руководство по написанию абстрактных планов”, книги *Руководство по настройке производительности*.

Примеры

Пример 1. Печать сообщения “yes”, если 3 больше 2:

```
if 3 > 2
    print "yes"
```

Пример 2. Условие if...else проверяет наличие авторов, почтовые индексы которых равны 94705. Если такие авторы есть, выводится строка “Berkeley author”:

```
if exists (select postalcode from authors
    where postalcode = "94705")
    print "Berkeley author"
```

Пример 3. Условие if...else проверяет наличие в базе данных объектов, созданных пользователями (идентификационные номера таких объектов превышают 100). Если в базе данных существуют таблицы, созданные пользователями, то инструкция else выдает сообщение об этом и выбирает их имена, типы и идентификационные номера.

```
if (select max(id) from sysobjects) < 100
    print "No user-created objects in this database"
else
 begin
    print "These are the user-created objects"
    select name, type, id
    from sysobjects
    where id > 100
 end
```

Пример 4. Поскольку в таблице titles значение объема общих продаж (total_sales) в строке с идентификатором PC9999 равно NULL, запрос возвращает значение FALSE. Часть запроса после ключевого слова else выполняется в том случае, когда часть запроса, которая стоит после ключевого слова if возвращает значение FALSE или NULL. Дополнительную информацию о логических выражениях и их значениях см. в разделе "Выражения" главы 4, "Выражения, идентификаторы и метасимволы".

```
if (select total_sales
    from titles
    where title_id = "PC9999") > 100
select "true"
else
select "false"
```

Использование

- Команда, следующая за ключевым словом if и относящимся к нему условием, выполняется при соблюдении условия (когда логическое выражение возвращает значение TRUE). После необязательного ключевого слова else идет оператор SQL, который будет выполнен при несоблюдении условия в инструкции if (когда логическое выражение возвращает значение FALSE).
- После ключевых слов if и else можно указать либо только один оператор SQL, либо блок операторов, заключенный между ключевыми словами begin и end (см. Пример 3).

  Блок операторов, идущих после ключевых слов if и else, может содержать команду execute и любые другие допустимые операторы SQL или блоки операторов.
- Если команда select используется как часть булева выражения, она должна возвращать единственное значение.
- Конструкции if...else могут использоваться либо в хранимых процедурах (где они часто применяются для проверки существования какого-либо параметра), либо в *нерегламентированных запросах* (см. примеры 1 и 2).

- Блок операторов, идущий после ключевых слов if или else, может содержать другую команду if (то есть допускаются вложения команд if). Максимальный уровень вложенности инструкций if зависит от сложности команд select (или других языковых конструкций), включенных в конструкцию if...else.

Примечание. Если в блоке if...else есть команда alter table, create table или create view , то до проверки условия создается схема для таблицы или представления. Это может привести к ошибкам, если таблица или представление уже существуют.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Конструкцию if...else по умолчанию могут использовать все пользователи. Для этого не требуется никаких полномочий.

См. также

Команды begin...end,create procedure

insert

Описание

Добавляет новые строки в таблицу или представление.

Синтаксис

```
insert [into]
[база_данных.[владелец.]]{имя_таблицы|имя_представления}
        [(список_столбцов)]
        {values (выражение [, выражение]...)
            |оператор_select [plan "абстрактный план"] }
```

Параметры

into

Необязательное ключевое слово.

имя_таблицы | *имя_представления*

Имя таблицы или представления, в которые добавляются новые строки. Если таблица или представление находятся в другой базе данных, то должно быть указано имя этой базы данных. Если в базе данных существует несколько таблиц или представлений с данным именем, то необходимо указать имя владельца. По умолчанию *владельцем* является текущий пользователь, а *базой_данных* – текущая база данных.

список_столбцов

Список, содержащий один или несколько столбцов, в которые будут добавляться данные. Этот список должен быть заключен в круглые скобки. Порядок столбцов в списке может быть любым, однако входные данные (перечисленные в инструкциях values или select) должны следовать в том же порядке. Если столбец обладает свойством IDENTITY, то фактическое имя столбца можно заменить ключевым словом syb_identity.

Список столбцов необходим тогда, когда данные заносятся лишь в некоторые из столбцов таблицы. Если список столбцов не задан, предполагается, что команда insert вставляет значения во все столбцы таблицы (в том порядке, в котором они были указаны в команде create table, создавшей эту таблицу).

Дополнительную информацию см. в разделе “Список столбцов” на стр. 595.

values

Ключевое слово, за которым следует список выражений.

выражение

Задает константы, переменные, параметры или значения NULL, которые будут вставлены в указанные столбцы. Символьные константы и константы, обозначающие дату/время, должны быть заключены в одинарные или двойные кавычки.

В качестве *выражения* нельзя использовать подзапрос.

Список значений должен быть заключен в круглые скобки и соответствовать явному или неявному списку столбцов. Дополнительную информацию о правилах ввода данных см. в разделе “Типы данных”.

оператор_select
Стандартная команда select, используемая для выборки значений, вставляемых в таблицу, из других таблиц или представлений.

plan "абстрактный план"
Определяет абстрактный план для оптимизации запроса. Это может быть полный или частичный план, заданный на языке абстрактных планов. Планы могут быть указаны только для команд insert...select. Дополнительную информацию см. в главе 30, “Руководство по написанию абстрактных планов”, книги *Руководство по настройке производительности*.

Примеры

Пример 1.

```
insert titles
values("BU2222", "Faster!", "business", "1389",
    null, null, null, "ok", "06/17/87", 0)
```

Пример 2.

```
insert titles
(title_id, title, type, pub_id, notes, pubdate,
    contract)
values ('BU1237', 'Get Going!', 'business',
    '1389', 'great', '06/18/86', 1)
```

Пример 3.

```
insert newauthors
    select *
    from authors
    where city = "San Francisco"
```

Пример 4.

```
insert test
    select *
    from test
    where city = "San Francisco"
```

Использование

- Команда insert используется только для добавления новых строк. Для изменения значений столбцов в уже существующих строках таблицы применяется команда update.

Список столбцов

- Список столбцов определяет порядок, в котором вводятся данные. Предположим, что таблица newpublishers по своей структуре и содержанию идентична таблице publishers в базе данных pubs2. В приведенном ниже примере список столбцов таблицы newpublishers совпадает со списком столбцов команды select, которая выбирает данные из таблицы publishers.

```
insert newpublishers (pub_id, pub_name)
select pub_id, pub_name
    from publishers
    where pub_name="New Age Data"
```

  Значения столбца pub_name, равные "New Age Data" и соответствующие им значения столбца pub_id заносятся в столбцы pub_id и pub_name таблицы newpublishers.

  В следующем примере порядок столбцов в списке столбцов таблицы newpublishers не совпадает с порядком столбцов в списке столбцов команды select, которая выбирает данные из таблицы publishers.

```
insert newpublishers (pub_id, pub_name)
    select pub_name, pub_id
    from publishers
    where pub_name="New Age Data"
```

  В результате значения столбца pub_id в строках со значением "New Age Data" будут занесены в столбец pub_name таблицы newpublishers, а значения pub_name в строках со значением "New Age Data" будут занесены в столбец pub_id этой таблицы.

- Элементы в списках столбцов и значений можно опускать, если соответствующие столбцы допускают значение null (см. Пример 2).

Проверка допустимости значений в столбцах

- Команда insert взаимодействует с параметрами ignore_dup_key, ignore_dup_row и allow_dup_row, установленными командой create index. Дополнительную информацию см. в описании команды create index.

- Область допустимых значений для столбца можно ограничить правилами и ограничениями check. Правила создаются командой create rule и назначаются столбцу системной процедурой sp_bindrule. Ограничения check объявляются в команде create table.

- Если значение не задано явным образом, то используется значение по умолчанию. Значения по умолчанию объявляются в команде create table или создаются командой create default и назначаются столбцу системной процедурой sp_bindefault.

- Если команда insert вставляет значения, которые нарушают условия допустимости или правила целостности (см. описания команд create rule и create trigger) или типы данных которых не соответствуют типам столбцов, указанным при создании таблицы (см. описание команды create table, а также главу 1, “Системные и пользовательские типы данных”), то эта команда не выполнится и будет выдано сообщение об ошибке.

Обработка пробелов

- При попытке вставить пустую строку (“”) в символьный столбец переменной длины или в столбец типа text будет вставлен один пробел. Столбцы типа char дополняются пробелами до заданной длины столбца.
- При вставке данных в столбцы типа varchar и univarchar все пробелы в конце строк удаляются, за исключением тех случаев, когда строка состоит только из пробелов. Строки, состоящие только из пробелов, усекаются до одного пробела. Если параметр string_rtruncation не установлен в on и длина вставляемой строки больше максимальной длины, заданной в определении столбца типа char, nchar, unichar, univarchar, varchar или nvarchar, то строка будет усечена без предупреждения.

Вставка данных в столбцы типов *text* и *image*

- Когда в столбцы типов text или image вставляется значение NULL, ни указатель на текст, ни текстовая страница не создаются. Чтобы получить указатель на текст для такого столбца, нужно выполнить команду update.

Триггеры команды *insert*

- Можно создать триггер, который будет совершать определенные действия при выполнении команды insert над заданной таблицей.

Использование команды *insert* при активированных службах Component Integration Services

- Команду insert можно направлять на удаленные серверы либо в виде запроса, содержащего текст этой команды на языке SQL, либо в качестве параметризованного динамического оператора.

Вставка строк из другой таблицы

- В одной команде можно выбирать строки из таблицы и вставлять их в ту же самую таблицу (см. Пример 4).
- При вставке данных, выбранных командой select из таблицы, имеющей в некоторых столбцах значения NULL, в таблицу, которая не допускает значений NULL, необходимо указать значения, которые бу-

дут подставлены вместо значений NULL исходной таблицы. Например, при вставке данных в таблицу advances, в которой не допускаются значения NULL, эти значения можно заменить нулями.

```
insert advances
select pub_id, isnull(advance, 0) from titles
```

Без применения функции isnull эта команда вставит все строки, не содержащие значения NULL, в таблицу advances, а для всех строк, в которых столбец advance таблицы titles имеет значение NULL, будут выданы сообщения об ошибке.

Если нельзя сделать такую подстановку данных, то нельзя будет и вставить данные, содержащие значения NULL, в столбцы, определенные как NOT NULL.

Две таблицы могут иметь одинаковую структуру, но отличаться друг от друга тем, что столбцы в одной таблице допускают значения NULL, а соответствующие им столбцы в другой таблице – не допускают. Узнать, для каких столбцов таблицы разрешены значения NULL, можно с помощью системной процедуры sp_help.

Команда *insert* и транзакции

- Если установлен режим связанных транзакций, то команда insert неявно начинает транзакцию, если нет текущей активной транзакции. Для завершения вставки необходимо зафиксировать транзакцию или выполнить откат изменений, например:

```
insert stores (stor_id, stor_name, city, state)
    values ('999', 'Books-R-Us', 'Fremont', 'AZ')
if exists (select t1.city
    from stores t1, stores t2
    where t1.city = t2.city
    and t1.state = t2.state
    and t1.stor_id < t2.stor_id)
        rollback transaction
else
        commit transaction
```

В режиме связанных транзакций этот пакет операторов начинает транзакцию и вставляет новую строку в таблицу stores. Если вставляемая строка имеет в столбцах city и state те же значения, что и какая-либо существующая строка этой таблицы, то производится откат внесенных в таблицу stores изменений, и транзакция завершается. В противном случае вставка фиксируется, и транзакция также завершается. Дополнительную информацию о режиме связанных транзакций см. в книге *Руководство пользователя Transact-SQL*.

Вставка значений в столбцы IDENTITY

- При вставке строки в таблицу имя столбца IDENTITY не должно включаться в список столбцов, а его значения – в список значений. Если таблица состоит только из одного столбца – столбца IDENTITY, – то список столбцов опускается, а список значений остается пустым, как показано в следующем примере:

```
insert id_table values()
```

- При первой вставке строки в таблицу столбцу IDENTITY присваивается значение 1. Для каждой новой строки значение в этом столбце увеличивается на единицу. Это значение имеет приоритет перед любыми значениями по умолчанию, объявленными для столбца в командах create table или alter table, либо назначенными ему при помощи системной процедуры sp_bindefault.

  При отказах сервера в последовательности значений столбцов IDENTITY могут возникать промежутки. Максимальный размер промежутка зависит от значения параметра конфигурации identity burning set factor. Промежутки также могут возникать в результате ручной вставки данных в столбец IDENTITY, удаления строк и откатов транзакций.

- Только владелец таблицы, владелец базы данных и системный администратор могут явным образом вставлять новое значение в столбец IDENTITY после того, как параметру identity_insert имя_таблицы было присвоено значение on для таблицы, содержащей этот столбец. Пользователь может за один раз установить параметр identity_insert имя_таблицы on только для одной таблицы в базе данных. Когда параметр identity_insert установлен в on, каждая команда insert должна содержать список столбцов и явно заданное значение для столбца IDENTITY.

  Вставка значения в столбец IDENTITY позволяет задавать начальное значение этого столбца или восстанавливать ошибочно удаленную строку. Если по столбцу IDENTITY не создан уникальный индекс, то уникальность вставляемого значения не проверяется; в столбец можно вставлять любое положительное целое число.

  Для явной вставки значения в столбец IDENTITY владелец таблицы, владелец базы данных или системный администратор должны сделать установку identity_insert имя_таблицы on для таблицы, содержащей этот столбец, а не для представления, через которое будет вставляться данное значение.

- Максимальное значение, которое можно вставить в столбец IDENTITY, равно 10 точность - 1. Когда столбец IDENTITY достигает этого значения, последующие команды insert возвращают сообщение об ошибке и текущая транзакция прерывается.

  В этом случае нужно с помощью команды create table создать новую таблицу, идентичную старой, но с более высокой точностью столбца IDENTITY, а затем с помощью команды insert или утилиты bcp скопировать данные из старой таблицы в новую.

- Глобальная переменная *@@identity* позволяет получать последнее значение, вставленное в столбец IDENTITY. Если последняя команда insert или select into была выполнена над таблицей, в которой нет столбца IDENTITY, то переменная *@@identity* возвращает значение 0.

- Для столбца IDENTITY, выбранного из одной таблицы и вставленного в другую, соблюдаются следующие правила наследования свойства IDENTITY:

  - Если столбец IDENTITY выбирается более одного раза, то в новой таблице он не допускает значений NULL. Такой столбец не наследует свойство IDENTITY.

  - Если столбец IDENTITY указан в списке выборки в составе выражения, то результирующий столбец не наследует свойство IDENTITY. Если хотя бы один из столбцов в выражении допускает значения NULL, то и новый столбец будет их допускать; в противном случае – не будет.

  - Если команда select содержит инструкцию group by или агрегатную функцию, столбец с агрегатной функцией не наследует свойство IDENTITY. Столбцы, содержащие агрегатную функцию над столбцом IDENTITY, будут допускать значения NULL; остальные – не будут.

  - Столбец IDENTITY, который вставляется в таблицу с объединением (оператором UNION) или соединением, не сохраняет свойство IDENTITY. Если таблица содержит объединение столбца IDENTITY и столбца, допускающего значения NULL, то новый столбец также допускает значения NULL; в противном случае этот столбец не допускает значений NULL.

Вставка данных через представления

- Если при создании представления указана инструкция with check option, то каждая вставляемая через это представление строка должна удовлетворять критериям отбора для этого представления.

Например, представление stores_cal включает все строки таблицы stores, у которых столбец state имеет значение “CA”.

```
create view stores_cal
as select * from stores
where state = "CA"
with check option
```

Инструкция with check option проверяет соответствие каждой команды insert критериям отбора данного представления. Строки, в которых столбец state не равен “CA”, не принимаются.

- Если при создании представления указана инструкция with check option, то все производные представления, созданные на его основе, должны удовлетворять критериям отбора для базового представления. Каждая строка, вставляемая через производное представление, должна быть видимой через базовое представление.

  Рассмотрим представление stores_cal30, созданное на основе представления stores_cal. Это новое представление содержит информацию о магазинах в Калифорнии с условиями платежа “Net 30”.

  ```
  create view stores_cal30
  as select * from stores_cal
  where payterms = "Net 30"
  ```

  Поскольку представление stores_cal было создано с применением инструкции with check option, все строки, вставляемые или обновляемые через представление stores_cal30, должны быть видимы через представление stores_cal. Строки, в которых значение столбца state не равно “CA”, не принимаются.

  Отметим, что представление stores_cal30 не имеет собственной инструкции with check option. Это означает, что через представление stores_cal30 можно вставлять или обновлять строки, в которых значение поля payterms не равно “Net 30”. Следующая команда update завершится успешно, хотя строка больше не будет видна через представление stores_cal30.

  ```
  update stores_cal30
  set payterms = "Net 60"
  where stor_id = "7067"
  ```

- Оператор insert запрещено выполнять над представлением соединения, созданным с инструкцией with check option.

- Если строка вставляется или обновляется через представление соединения, все изменяемые столбцы должны принадлежать одной и той же базовой таблице.

Секционирование таблиц для повышения производительности операций вставки

- Несекционированная таблица без кластерного индекса состоит из цепочки страниц базы данных, причем каждая из страниц связана с предшествующей и с последующей. Поэтому каждая операция вставки использует последнюю страницу в этой цепочке. Во время операции вставки удерживается монопольная блокировка последней страницы. Следовательно, другие параллельно выполняющиеся транзакции не могут вставлять данные в таблицу.

  При секционировании таблицы при помощи команды alter table с инструкцией partition создаются дополнительные цепочки страниц. Каждая цепочка имеет собственную последнюю страницу, которая может использоваться для параллельных операций вставки. Это повышает производительность благодаря снижению конкуренции за страницы. Если таблица распределена по нескольким физическим устройствам, производительность операций вставки повышается при секционировании также за счет снижения числа конфликтов ввода-вывода в то время, когда сервер сбрасывает данные из кэша на диск. Дополнительную информацию о секционировании таблиц для повышения производительности операций вставки см. в книге *Руководство по настройке производительности*.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Ниже приведен список расширений Transact-SQL:

- оператор union в части select команды insert;
- уточнение имени таблицы или столбца именем базы данных;
- вставка данных через представление, которое содержит соединение.

Примечание. В режиме FIPS flagger операции вставки через представление, содержащее соединение, не распознаются.

Полномочия

- Полномочия на выполнение команды insert по умолчанию принадлежат владельцу таблицы или представления, который может передавать эти полномочия другим пользователям.
- Полномочия на выполнение команды insert для столбца IDENTITY в таблице принадлежат владельцу таблицы, владельцу базы данных и системному администратору.

См. также

Команды alter table, create default, create index, create rule, create table, create trigger, dbcc, delete, select, update

Типы данных Глава 1, “Системные и пользовательские типы данных”

Системные процедуры sp_bindefault, sp_bindrule, sp_help, sp_helpartition, sp_unbindefault, sp_unbindrule

Утилиты bcp

kill

Описание — Завершает процесс.

Синтаксис — kill *spid*

Параметры — *spid*

Идентификационный номер процесса, который требуется завершить. Значение переменной *spid* должно быть константой; его нельзя передавать в хранимую процедуру как параметр или использовать как локальную переменную. Список процессов и другую необходимую информацию можно получить при помощи системной процедуры sp_who .

Примеры

```
kill 1378
```

Использование

- Отчет о текущих процессах можно получить с помощью системной процедуры sp_who. Ниже приведен типичный отчет.

```
fid spid  status    loginame origname hostname blk  dbname
      cmd
--- ----- --------   -------- -------- -------- --- ------
      --------------
  0    1  recv sleep  bird     bird     jazzy    0   master
      AWAITING COMMAND
  0    2  sleeping    NULL     NULL             0   master
      NETWORK HANDLER
  0    3  sleeping    NULL     NULL             0   master
      MIRROR HANDLER
  0    4  sleeping    NULL     NULL             0   master
      AUDIT PROCESS
  0    5  sleeping    NULL     NULL             0   master
      CHECKPOINT SLEEP
  0    6  recv sleep  rose     rose     petal    0   master
      AWAITING COMMAND
  0    7  running     robert   sa       helos    0   master
      SELECT
  0    8  send sleep  daisy    daisy    chain    0   pubs2
      SELECT
  0    9  alarm sleep lily     lily     pond     0   master
      WAITFOR
  0   10  lock sleep  viola    viola    cello    7   pubs2
      SELECT
```

Идентификационные номера процессов, используемые в команде kill языка Transact-SQL содержатся в столбце spid. Столбец blk содержит идентификационный номер блокирующего процесса, если таковой существует. Блокирующий процесс – это процесс, занимающий

ресурсы, необходимые другим процессам. Блокирующий процесс может удерживать монопольную блокировку. В нашем примере процесс 10 (выполняющий команду select для таблицы) заблокирован процессом 7 (выполняющим команды begin transaction и insert для той же таблицы).

- Столбец статус отражает состояние процесса. В следующей таблице перечислены состояния процессов и действие команды kill.

Таблица 7-26. Статус процессов, отображаемый системной процедурой sp_who

Статус	Описание	Когда выполняется действие, предписанное командой kill
`recv sleep`	Ожидание приема данных по сети.	Немедленно.
`send sleep`	Ожидание передачи данных по сети.	Немедленно.
`alarm sleep`	Ожидание сигнала таймера, например waitfor delay “10:00".	Немедленно.
`lock sleep`	Ожидание получения блокировки.	Немедленно.
`sleeping`	Ожидание дискового ввода-вывода или другого ресурса. Означает, что процесс, вероятно, выполняется, но осуществляет большой объем операций дискового ввода-вывода.	Процесс будет завершен, как только он “проснется”. Обычно это происходит сразу, однако некоторые уснувшие процессы больше не просыпаются, и для их удаления необходима перезагрузка сервера Adaptive Server.
`runnable`	Процесс находится в очереди готовых к выполнению процессов.	Немедленно.
`running`	Процесс активно выполняется в одном из экземпляров сервера.	Немедленно.
`infected`	Сервер Adaptive Server обнаружил серьезную ошибку; случается очень редко.	Команду kill применять не рекомендуется. Для удаления процесса, вероятно, потребуется перезапуск сервера Adaptive Server.
`background`	Процесс (например, процедура обработки события, связанного с достижением порогового значения для свободного места в сегменте журнала) запущен сервером Adaptive Server, а не процессом пользователя.	Немедленно; команду kill нужно использовать с особой осторожностью. Рекомендуется тщательно проверить таблицу sysprocesses перед завершением фонового процесса.
`log suspend`	Процессы приостановлены из-за достижения последнего порога, установленного для свободного места в сегменте журнала	Немедленно

- Для получения отчета о текущих блокировках и номерах *spid* процессов, которые их удерживают, используется системная процедура sp_lock.

Стандарты	Уровень соответствия стандарту SQL92: расширение Transact-SQL.
Полномочия	Полномочия на выполнение команды kill по умолчанию принадлежат системному администратору и не могут быть переданы другим пользователям.
См. также	**Команды** shutdown
	Системные процедуры sp_lock, sp_who

load database

Описание

Эта команда загружает резервную копию пользовательской базы данных, включая ее журнал транзакций, созданный командой dump database.

Синтаксис

```
load database имя_базы_данных
        from [compress::]устройство_с_чередованием
            [at имя_сервера_backup_server ]
            [density = значение_плотности,
            blocksize = число_байтов,
            dumpvolume = имя_тома,
            file = имя_файла]
        [stripe on [compress::]устройство_с_чередованием
            [at имя_сервера_backup_server ]
            [density = значение_плотности,
            blocksize = число_байтов,
            dumpvolume = имя_тома,
            file = имя_файла]
        [[stripe on [compress::]устройство_с_чередованием
            [at имя_сервера_backup_server ]
            [density = значение_плотности,
            blocksize = число_байтов,
            dumpvolume = имя_тома,
            file = имя_файла]]...]
        [with {
            density = значение_плотности,
            blocksize = число_байтов,
            dumpvolume = имя_тома,
            file = имя_файла,
            [dismount | nodismount],
            [nounload | unload],
            listonly [= full],
            headeronly,
            notify = {client | operator_console}
            }]]
```

Параметры

имя_базы_данных

Имя базы данных, которая будет принимать резервную копию. Она может быть или базой данных, созданной с параметром for load, или существующей базой данных. Резервные копии данных при загрузке в существующую базу данных заменяют все существующие данные. Размер принимающей базы данных должен быть не меньше размера зарезервированной базы данных. Имя базы данных может быть задано в виде литерала, локальной переменной или параметра хранимой процедуры.

compress::

Вызывает восстановление сжатой архивной базы данных. Дополнительную информацию о параметре compress см. в главе 27, “Резервное копирование и восстановление пользовательских баз данных”, книги *Руководство по системному администрированию*.

from устройство_с_чередованием
Устройство, с которого загружаются данные. Дополнительную информацию о том, какую форму использовать при указании устройства для хранения резервных копий, см. в разделе “Указание устройств для хранения резервных копий” на стр. 622. Список поддерживаемых устройств для хранения резервных копий см. в Руководстве по конфигурации и установке сервера Adaptive Server.

at имя_сервера_backup_server
Имя удаленного сервера Backup Server, работающего на машине, к которой присоединено устройство для хранения резервных копий. Если операционная система использует файлы интерфейсов, *имя_сервера_backup_server* должно содержаться в одном из этих файлов.

density = значение_плотности
Игнорируется. Дополнительную информацию см. в описании команды dump database.

blocksize = число_байтов
Переопределяет размер блока по умолчанию для устройства хранения резервных копий. Если размер блока указывается в системе UNIX, то он должен совпадать с размером, использовавшимся при резервном копировании. Дополнительную информацию см. в описании команды dump database.

dumpvolume = имя_тома
Поле имени тома в метке ленты в стандарте ANSI. Команда load database проверяет эту метку при открытии ленты и формирует сообщение об ошибке, если загружается неверный том.

Примечание. При использовании команды load database параметр dumpvolume не создает сообщений об ошибках, если для параметра file=*имя_файла* задано неправильное имя файла. Сервер Backup Server выполняет поиск этого файла по всей ленте, несмотря на то что смонтирована неправильная лента.

---

stripe on устройство_с_чередованием
Дополнительное устройство для хранения резервных копий. Можно использовать до 32 устройств, включая устройства, названные в инструкции to *устройство_с_чередованием*. Сервер Backup Server загружает данные со всех устройств одновременно, уменьшая необходимые для этого время и число изменений томов. Информацию о том, как указывать устройство для хранения резервных копий, см. в разделе “Указание устройств для хранения резервных копий” на стр. 622.

dismount | nodismount

На платформах, поддерживающих логический демонтаж, определяет, останутся ли ленты смонтированными. По умолчанию, все ленты, использовавшиеся для загрузки, после ее завершения демонтируются. При использовании параметра nodismount ленты остаются доступными для дополнительных загрузок или создания дополнительных резервных копий.

nounload | unload

Определяет, перематываются ли ленты на начало после завершения загрузки. По умолчанию ленты не перематываются на начало, что позволяет выполнять дополнительные загрузки из одного и того же тома ленты. Параметр unload указывается для последнего файла резервной копии, загружаемого из тома с несколькими резервными копиями. В этом случае после завершения загрузки лента перематывается на начало и выгружается.

file = имя_файла

Имя отдельной резервной копии базы данных в томе ленты. Если во время создания резервных копий имена файлов этих резервных копий не были записаны, для отображения информации обо всех файлах резервных копий используется параметр listonly.

listonly [= full]

Отображает информацию обо всех файлах резервных копий в томе ленты, но *не загружает базу данных*. Параметр listonly идентифицирует базу данных и устройство, дату и время создания резервной копии, а также дату и время, когда она может быть переписана. При значении параметра listonly = full данный параметр обеспечивает дополнительную информацию о резервной копии. Оба отчета сортируются по метке ленты в стандарте ANSI.

После вывода списка файлов в томе сервер Backup Server посылает запрос на изменение тома. Оператор может или смонтировать другой том ленты, или завершить операцию вывода списков для всех устройств хранения резервных копий.

В текущей версии параметр listonly переопределяет параметр headeronly.

Предупреждение. Команду load database with listonly нельзя выполнять для картриджей с лентой шириной 1/4 дюйма.

---

headeronly

Отображает информацию заголовка для одного файла резервной копии, но *не загружает базу данных*. Параметр headeronly отображает информацию о первом файле на ленте, если для указания имени другого файла не используется параметр file = *имя_файла*. В заголовке файла резервной копии указываются:

- Тип резервной копии (база данных или журнал транзакций)
- Идентификатор базы данных
- Имя файла
- Дата создания резервной копии
- Набор символов
- Порядок сортировки
- Количество страниц
- Идентификатор следующего объекта

notify = {client | operator_console}

Переопределяет получателя сообщения по умолчанию.

- В операционных системах, в которых допускается использование терминалов операторов, сообщения об изменениях томов всегда посылаются на терминал оператора, находящийся на машине сервера Backup Server. Для направления других сообщений сервера Backup Server в сеанс работы с терминалом, в котором была инициирована команда dump database, используется параметр client.
- В операционных системах (например, UNIX), в которых не допускается использование терминалов операторов, сообщения посылаются клиенту, инициировавшему команду dump database. Для направления сообщений на терминал, находящийся на машине сервера Backup Server, используется параметр operator_console.

Примеры

Пример 1. Повторная загрузка базы данных pubs2 с ленточного устройства:

```
load database pubs2
    from "/dev/nrmt0"
```

Пример 2. Загрузка базы данных pubs2 с использованием сервера Backup Server с именем REMOTE_BKP_SERVER. В этой команде указываются три устройства:

```
load database pubs2
        from "/dev/nrmt4" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt0" at REMOTE_BKP_SERVER
```

Пример 3. Загрузка, базы данных pubs2 из сжатого файла резервной копии с именем *dmp090100.dmp*, расположенного в каталоге */opt/bin/Sybase/dumps*:

```
load database pubs2 from
    "compress::/opt/bin/Sybase/dumps/dmp090100.dmp"
```

Использование

- Параметры listonly и headeronly отображают информацию о файлах резервных копий без их загрузки.
- Резервное копирование и загрузка выполняются через сервер Backup Server.
- Чтобы убедиться в том, что базы данных правильно синхронизированы, то есть схемы всех прокси-таблиц синхронизированы с содержанием повторно загруженной первичной базы данных, может потребоваться выполнение команды alter database имя_бд for proxy_update на том сервере, где размещена прокси-БД.
- В таблице 7-27 описываются команды и системные процедуры, используемые для восстановления баз данных из резервных копий.

Таблица 7-27. Команды, используемые для восстановления баз данных из резервных копий

Команда	Предназначение
create database for load	Создание базы данных для загрузки резервной копии
load database	Восстановление базы данных из резервной копии
load transaction	Применение последних транзакций к восстановленной базе данных
online database	Обеспечение доступности базы данных для публичного использования после обычной последовательности загрузки или после обновления базы данных до текущей версии сервера Adaptive Server
load { database | transaction } with {headeronly | listonly}	Идентификация файлов резервных копий на ленте
sp_volchanged	Ответ на сообщения сервера Backup Server об изменениях томов

Ограничения

- Нельзя загружать резервную копию, созданную на другой платформе.
- Нельзя загружать резервную копию, созданную на сервере версии ниже чем 10.0.
- Если на базу данных распространяются межбазовые ограничения ссылочной целостности, в системной таблице sysreferences хранится *имя*, а не идентификационный номер внешней базы данных. Сервер Adaptive Server не может гарантировать ссылочную целостность, если для изменения имени базы данных или ее загрузки на другой сервер используется команда load database.

- При каждом добавлении или удалении межбазового правила целостности либо удалении таблицы, на которую распространяется межбазовое правило целостности, необходимо выполнять резервное копирование *обеих* затронутых баз данных.

  Предупреждение. Загрузка более ранних резервных копий этих баз данных может вызвать повреждение базы данных. Перед резервным копированием базы данных для ее загрузки с другим именем или перемещения на другой сервер Adaptive Server необходимо использовать команду alter table, чтобы удалить все внешние ограничения ссылочной целостности.

- Команда load database удаляет подозрительные записи страниц, относящиеся к загруженной базе данных, из таблицы master..sysattributes.
- Команда load database переписывает все данные, существующие в базе данных.
- После загрузки резервной копии базы данных может потребоваться дополнительное время на выполнение двух процессов перед переводом базы данных в оперативный режим.
  - После завершения загрузки все неиспользуемые страницы в базе данных нужно обнулить. Необходимое для этого время зависит от количества неиспользуемых страниц. Если размер целевой и загружаемой базы данных одинаков, этот шаг выполняется сервером Backup Server. Если целевая база данных больше, чем загружаемая база данных, этот шаг выполняется сервером Adaptive Server после завершения загрузки сервером Backup Server. Необходимое для этого шага время зависит от количества пустых страниц.
  - Для всех транзакций из журнала транзакций, включенного в резервную копию базы данных, должны быть выполнены откат или повтор этих транзакций. Необходимое для этого время зависит от количества и типа транзакций в журнале. Этот шаг выполняется сервером Adaptive Server.
- Размер принимающей базы данных должен быть не меньше размера загружаемой базы данных. Если размер принимающей базы данных слишком мал, сервер Adaptive Server выдает сообщение об ошибке, в котором указывается необходимый размер.
- Нельзя выполнять загрузку с устройства типа NULL (в UNIX – /dev/null).

- Команду load database нельзя использовать в пользовательской транзакции.

Блокировка пользователей во время загрузок

- Во время загрузки базы данных ее нельзя использовать. Команда load database устанавливает статус базы данных как “автономный”. Нельзя пользоваться базой данных со статусом “автономный”. “Автономный” статус запрещает пользователям доступ к базе данных и ее изменение в течение загрузки.
- База данных, загруженная с помощью команды load database, остается недоступной до выполнения команды online database.

Обновление резервных копий баз данных и журналов транзакций

- Для восстановления и обновления резервной копии базы данных пользователя с версии сервера 10.0 или более поздней до текущей версии сервера Adaptive Server выполняются следующие действия.

  a Загружается самая последняя резервная копия, базы данных.

  b Загружаются (*по порядку*) все резервные копии журналов транзакций, созданные с момента последнего резервирования базы данных.

  Сервером Adaptive Server проверяется временная метка каждой резервной копии, чтобы обеспечить ее загрузку в соответствующую базу данных и в правильной последовательности.

  c Выполняется команда online database, чтобы выполнить обновление и сделать базу данных доступной для публичного использования.

  d Делается резервная копия обновленной базы данных сразу после обновления, чтобы эта копия была согласована с текущей версией сервера Adaptive Server.

Указание устройств для хранения резервных копий

- Устройство для хранения резервных копий можно указывать в качестве литерала, локальной переменной или параметра хранимой процедуры.
- Имя локального устройства можно задавать как:
  - имя логического устройства из системной таблицы sysdevices;
  - полное имя пути;
  - относительное имя пути.

  Относительные имена пути разрешаются сервером Backup Server с использованием текущего рабочего каталога сервера Adaptive Server.

- При загрузке через сеть указывается полное имя пути к устройству для хранения резервных копий. Имя пути должно быть допустимым на машине сервера Backup Server. Если имя включает символы, отличные от букв, чисел или символа подчеркивания (_), это имя целиком заключается в кавычки.
- На применение команд загрузки могут влиять проблемы владения устройством для хранения резервных копий и полномочий на его использование.
- Можно делать одновременно несколько загрузок (или резервных копий), если загрузки осуществляются с разных физических устройств.

Серверы Backup Server

- Сервер Backup Server должен работать на той же машине, что и сервер Adaptive Server. Сервер Backup Server должен быть указан в таблице master..sysservers. Эта запись создается во время установки или обновления и не должна удаляться.
- Если устройства для резервирования расположены на другой машине и загрузка происходит через сеть, на удаленной машине также должен быть установлен сервер Backup Server.

Имена томов

- Тома резервных копий помечаются в соответствии со стандартом ANSI на метки лент. Метка включает номер логического тома и позицию устройства в наборе полос.
- Во время загрузок сервер Backup Server использует метку ленты для проверки правильности порядка, в котором монтируются тома. Это позволяет производить загрузки с меньшего количества устройств, чем использовалось при создании резервных копий.

Примечание. При создании резервных копий и выполнении загрузок через сеть для каждой операции необходимо указывать одинаковое количество устройств с чередованием сегментов/разделов.

Замена томов резервных копий

- Если сервер Backup Server обнаруживает проблему со смонтированным в данный момент томом, он посылает запрос на замену тома либо клиенту, либо на его консоль оператора. Когда другой том смонтирован, оператор должен уведомить об этом сервер Backup Server, выполнив системную процедуру sp_volchanged на любом сервере Adaptive Server, взаимодействующем с сервером Backup Server.

Восстановление системных баз данных

- Пошаговые инструкции по восстановлению системных баз данных из резервных копий см. в книге *Руководство по системному администрированию*.

Зеркальное копирование дисков

- В начале загрузки сервер Adaptive Server передает серверу Backup Server имя первичного устройства для каждого устройства логической базы данных и каждого устройства логического журнала. Если режим зеркального копирования для первичного устройства отключен, сервер Adaptive Server вместо этого передает имя вторичного устройства. Если с любым из названных устройств происходит сбой до завершения передачи данных, загрузка аварийно прекращается.
- Если попытаться отключить режим зеркального копирования для любого названного устройства во время выполнения команды load database, сервер Adaptive Server выдает сообщение. Пользователь, выполняющий команду disk unmirror, может прекратить загрузку или отложить выполнение команды disk unmirror до завершения загрузки.
- Сервер Backup Server загружает данные в первичное устройство, затем командой load database копирует их во вторичное устройство. Команда load database требует больше времени для завершения, если какое-либо устройство для хранения базы данных зеркалировано.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду load database могут выполнять только системный администратор, владелец базы данных и пользователь с ролью Operator.

См. также

Команды alter database, dbcc, dump database, dump transaction, load transaction, online database

Системные процедуры sp_helpdevice, sp_volchanged, sp_helpdb

load transaction

Описание

Загружает резервную копию журнала транзакций, созданного командой dump transaction.

Синтаксис

load tran[saction] *имя_базы_данных*
from [compress::]*устройство_с_чередованием*
[at *имя_сервера_backup_server*]
[density = *значение_плотности*,
blocksize = *число_байтов*,
dumpvolume = *имя_тома*,
file = *имя_файла*]
[stripe on [compress::]*устройство_с_чередованием*
[at *имя_сервера_backup_server*]
[density = *значение_плотности*,
blocksize = *число_байтов*,
dumpvolume = *имя_тома*,
file = *имя_файла*]
[[stripe on [compress::]*устройство_с_чередованием*
[at *имя_сервера_backup_server*]
[density = *значение_плотности*,
blocksize = *число_байтов*,
dumpvolume = *имя_тома*,
file = *имя_файла*]]...]
[with {
density = *значение_плотности*,
blocksize = *число_байтов*,
dumpvolume = *имя_тома*,
file = *имя_файла*,
[dismount | nodismount],
[nounload | unload],
listonly [= full],
headeronly,
notify = {client | operator_console}
until_time = *дата_время*}]]

Параметры

имя_базы_данных

Имя базы данных, которая будет получать данные из резервной копии журнала транзакций. Сегмент журнала принимающей базы данных должен быть не меньше, чем сегмент журнала из резервной копии базы данных. В качестве имени базы данных может быть указан литерал, локальная переменная или параметр хранимой процедуры.

compress::

Вызывает восстановление сжатого архивного журнала транзакций. Дополнительную информацию о параметре compress см. в главе 27, “Резервное копирование и восстановление пользовательских баз данных”, книги *Руководство по системному администрированию*.

from устройство_с_чередованием
Имя устройства для хранения резервных копий, из которого загружается журнал транзакций. Информацию о том, какую форму использовать при указании устройства для хранения резервных копий, см в разделе “Указание устройств для хранения резервных копий” на стр. 622. Список поддерживаемых устройств для хранения резервных копий см. в руководствах по конфигурации и установке сервера Adaptive Server.

at имя_сервера_backup_server
Имя удаленного сервера Backup Server, работающего на машине, к которой присоединено устройство для хранения резервных копий. *Имя_сервера_backup_server* должно содержаться в файле интерфейсов операционной системы (если он используется).

density = значение_плотности
Переопределяет значение плотности по умолчанию для ленточного устройства. *Этот параметр игнорируется.*

blocksize = число_байтов
Переопределяет размер блока по умолчанию для устройства хранения резервных копий. Если размер блока указывается в системе UNIX, он должен совпадать с размером, использовавшимся при резервном копировании.

dumpvolume = имя_тома
Поле имени тома в метке ленты в стандарте ANSI. Команда load transaction проверяет эту метку при открытии ленты и формирует сообщение об ошибке, если загружается неверный том.

stripe on устройство_с_чередованием
Дополнительное устройство для хранения резервных копий. Можно использовать до 32 устройств, включая устройства, названные в инструкции to *устройство_с_чередованием*. Сервер Backup Server загружает данные со всех устройств одновременно, уменьшая необходимые для этого время и число изменений томов. Информацию о том, как указывать устройство для хранения резервных копий, см. в разделе “Указание устройств для хранения резервных копий” на стр. 622.

dismount | nodismount
На платформах, поддерживающих логический демонтаж, определяет, останутся ли ленты смонтированными. По умолчанию все ленты, использовавшиеся для загрузки, после ее завершения демонтируются. При использовании параметра nodismount ленты остаются доступными для дополнительных загрузок или создания дополнительных резервных копий.

nounload | unload

Определяет, перематываются ли ленты на начало после завершения загрузки. По умолчанию ленты не перематываются на начало, что позволяет выполнять дополнительные загрузки из одного и того же тома ленты. Параметр unload указывается для последнего файла резервной копии, загружаемого из тома с несколькими резервными копиями. В этом случае после завершения загрузки лента перематывается на начало и выгружается.

file = имя_файла

Имя отдельной резервной копии базы данных в томе ленты. Если во время создания резервных копий имена файлов этих резервных копий не были записаны, для отображения информации обо всех файлах резервных копий используется параметр listonly.

listonly [= full]

Отображает информацию обо всех файлах резервных копий в томе ленты, но *не загружает журнал транзакций*. Параметр listonly идентифицирует базу данных и устройство, дату и время создания резервной копии, а также дату и время, когда она может быть переписана. При значении параметра listonly = full данный параметр обеспечивает дополнительную информацию о резервной копии. Оба отчета сортируются по метке ленты в стандарте ANSI.

После вывода списка файлов в томе сервер Backup Server посылает запрос на изменение тома. Оператор может или смонтировать другой том ленты, или завершить операцию вывода списков для всех устройств хранения резервных копий.

В текущей версии параметр listonly переопределяет параметр headeronly.

Предупреждение. Команду load transaction with listonly для картриджей с лентой шириной 1/4 дюйма выполнять нельзя.

---

headeronly

Отображает информацию заголовка для одного файла резервной копии, но *не загружает базу данных*. Параметр headeronly отображает информацию о первом файле на ленте, если для указания имени другого файла не используется параметр file = имя_файла. В заголовке файла резервной копии указываются:

- тип резервной копии (база данных или журнал транзакций);
- идентификатор базы данных;
- имя файла;

- дата создания резервной копии;
- набор символов;
- порядок сортировки;
- количество страниц;
- идентификатор следующего объекта;
- местоположение контрольных точек в журнале;
- местоположение самой первой по времени записи begin transaction;
- старые и новые даты последовательностей.

notify = {client | operator_console}

Переопределяет получателя сообщения по умолчанию.

- В операционных системах, в которых допускается использование терминалов операторов, сообщения об изменениях томов всегда посылаются на терминал оператора, находящийся на машине сервера Backup Server. Для направления других сообщений сервера Backup Server в сеанс работы с терминалом, в котором была инициирована команда dump database, используется параметр client.
- В операционных системах (например, UNIX), в которых не допускается использование терминалов операторов, сообщения посылаются клиенту, инициировавшему команду dump database. Для направления сообщений на терминал, находящийся на машине сервера Backup Server, используется параметр operator_console.

until_time

Загружает журнал транзакций по состоянию на конкретный момент времени, указанный в журнале транзакций. В базе данных сохраняются только те транзакции, которые были зафиксированы до указанного времени.

Примеры

Пример 1. Загрузка журнала транзакций в базу данных pubs2 с ленточного устройства:

```
load transaction pubs2
    from "/dev/nrmt0"
```

Пример 2. Загрузка журнала транзакций в базу данных pubs2 с использованием сервера Backup Server с именем REMOTE_BKP_ SERVER.

```
load transaction pubs2
        from "/dev/nrmt4" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt5" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt0" at REMOTE_BKP_SERVER
```

Пример 3. Загрузка журнала транзакций в базу данных pubs2 по состоянию на 10:51:43:866 утра 20 марта 1997 года:

```
load transaction pubs2
    from "/dev/ntmt0"
    with until_time = "mar 20, 1997 10:51:43:866am"
```

Использование

- Параметры listonly и headeronly отображают информацию о файлах резервных копий без их загрузки.
- Резервное копирование и загрузка выполняются через сервер Backup Server.
- В таблице 7-28 описываются команды и системные процедуры, используемые для восстановления баз данных из резервных копий:

Таблица 7-28. Команды, используемые для восстановления баз данных

Команда	Предназначение
create database for load	Создание базы данных для загрузки резервной копии.
load database	Восстановление базы данных из резервной копии.
load transaction	Применение последних транзакций к восстановленной базе данных.
online database	Обеспечение доступности базы данных для публичного использования после обычной последовательности загрузки или после обновления базы данных до текущей версии сервера Adaptive Server.
load { database | transaction } with {headeronly | listonly}	Идентификация файлов резервных копий на ленте
sp_volchanged	Ответ на сообщения сервера Backup Server об изменениях томов.

Ограничения

- Нельзя загружать резервную копию, созданную на другой платформе.
- Нельзя загружать резервную копию, созданную на сервере версии ниже чем 10.0
- Версии базы данных и журналов транзакций должны быть одинаковы.
- Журналы транзакций необходимо загружать в хронологическом порядке.
- Нельзя выполнять загрузку с устройства типа NULL (в UNIX – /dev/null).
- Нельзя выполнять команду load transaction после команды online database, которая обновляет базу данных до новой версии Adaptive Server. Чтобы обновить базу данных до новой версии, команды

должны идти в следующей последовательности – load database, load transaction и online database.

- Команду online database нельзя выполнять до того, как будут загружены все журналы транзакций. Последовательность команд должна быть следующей:

  a Load database

  b Load transaction (этих команд может быть несколько)

  c Online database

  Однако чтобы загрузить дополнительные журналы транзакций, оставив базу данных доступной только для чтения (типичная ситуация "теплого резервирования"), нужно выполнить команду dump tran with standby_access для создания резервных копий транзакций. Затем можно будет выполнить команду online database for standby_access, чтобы разрешить доступ только для чтения к базе данных.

- Команду load transaction нельзя выполнять в пользовательской транзакции.

Восстановление базы данных

- Для восстановления базы данных выполняются следующие действия:
  - загружается самая последняя резервная копия базы данных;
  - загружаются (*по порядку*) все резервные копии журналов транзакций, созданные с момента последнего резервирования базы данных;
  - выполняется команда online database, чтобы сделать базу данных доступной для публичного использования.
- При каждом добавлении или удалении межбазового правила целостности либо удалении таблицы, на которую распространяется межбазовое правило целостности, необходимо выполнять резервное копирование *обеих* затронутых баз данных.

Предупреждение. Загрузка более ранних резервных копий этих баз данных может вызвать повреждение базы данных.

- Дополнительную информацию о резервном копировании и восстановлении баз данных на сервере Adaptive Server см. в книге *Руководство по системному администрированию*.

Восстановление базы данных на определенный момент времени

- Можно использовать параметр until_time для большинства загружаемых или резервируемых баз данных. Он не применим к таким базам данных, как master, в которых данные и журналы находятся на одном и том же устройстве. Также нельзя использовать его для любой базы данных, журнал которой был очищен после последнего выполнения команды dump database, например tempdb.
- Применение параметра until_time полезно по следующим причинам.
  - Он обеспечивает согласованность базы данных на конкретный момент времени. Например, в среде баз данных с системой поддержки принятия решений (DSS) и оперативной обработки транзакций (OLTP) системный администратор может выполнять откат базы данных с DSS к более раннему указанному моменту времени, чтобы сравнивать данные между более ранней и текущей версиями.
  - Если пользователь непреднамеренно уничтожил данные, например удалил важную таблицу, можно использовать параметр until_time, чтобы отменить ошибочную команду посредством повтора всех транзакций до момента времени перед уничтожением данных.
- Чтобы эффективно использовать параметр until_time в случае уничтожения данных, необходимо знать точное время ошибки. Его можно найти, выполнив команду select getdate() сразу после возникновения ошибки. Для нахождения более точного значения времени (до миллисекунд) используется функция convert, например:

```
select convert(char(26), getdate(), 109)
--------------------------
Feb 26 1997 12:45:59:650PM
```

- После загрузки журнала транзакций с использованием параметра until_time сервер Adaptive Server перезапускает последовательность журналов базы данных. Это означает, что до того, как будет выполнено следующее резервирование базы данных, после выполнения команды load transaction с параметром until_time последующие журналы транзакций загружать нельзя. Необходимо создать резервную копию базы данных перед созданием резервной копии следующего журнала транзакций.
- В базе данных сохраняются только те транзакции, которые были зафиксированы до указанного времени. Однако в некоторых случаях к данным базы данных применяются транзакции, зафиксированные вскоре после указанного в параметре until_time времени. Это может

произойти, если в одно и то же время выполняется фиксация нескольких транзакций. В журнале транзакций эти транзакции могут оказаться упорядоченными не по времени. В этом случае транзакции, оказавшиеся за пределами временной последовательности, отражаются в восстановленных данных. Временной промежуток должен быть меньше секунды.

- Дополнительную информацию о восстановлении базы данных на определенный момент времени см. в книге *Руководство по системному администрированию*.

Блокировка пользователей во время загрузок

- При загрузке базы данных использовать ее нельзя. Команда load transaction в отличие от команды load database не изменяет автономный или оперативный статус базы данных. Команда load transaction оставляет статус базы данных неизмененным. Команда load database устанавливает статус базы данных как “автономный”. Нельзя пользоваться базой данных со статусом “автономный”. “Автономный” статус запрещает пользователям доступ к базе данных и ее изменение в течение загрузки.
- База данных, загруженная с помощью команды load database, остается недоступной до выполнения команды online database.

Обновление резервных копий баз данных и журналов транзакций

- Для восстановления и обновления резервной копии базы данных пользователя с версии сервера 10.0 или более поздней до текущей версии сервера Adaptive Server выполняются следующие действия:
  a загружается самая последняя резервная копия базы данных;
  b загружаются (*по порядку*) все журналы транзакций, созданные с момента последнего резервирования базы данных;
  c выполняется обновление с помощью команды online database;
  d делается резервная копия обновленной базы данных сразу после обновления, чтобы эта копия была согласована с текущей версией сервера Adaptive Server.

Указание устройств для хранения резервных копий

- Устройство для хранения резервных копий можно указывать в качестве литерала, локальной переменной или параметра хранимой процедуры.

- При загрузке с локального устройства можно задавать имя устройства для хранения резервных копий как:
  - полное имя пути;
  - относительное имя пути;
  - имя логического устройства из системной таблицы sysdevices.

  Относительные имена пути разрешаются сервером Backup Server с использованием текущего рабочего каталога сервера Adaptive Server.
- При загрузке через сеть указывается полное имя пути к устройству для хранения резервных копий. (Нельзя использовать относительное имя пути или имя логического устройства из системной таблицы sysdevices.) Имя пути должно быть допустимым на машине сервера Backup Server. Если имя включает символы, отличные от букв, чисел или символа подчеркивания (_), его необходимо заключать в кавычки.
- На применение команд загрузки могут влиять проблемы владения устройством для хранения резервных копий и полномочий на его использование. Команда sp_addumpdevice добавляет устройство к системным таблицам, но не гарантирует возможность загрузки с этого устройства или создания файла как устройства для хранения резервных копий.
- Можно делать одновременно несколько загрузок (или резервных копий), если эти загрузки осуществляются с разных физических устройств.

Сервер Backup Server

- Сервер Backup Server должен работать на той же машине, что и сервер Adaptive Server. Сервер Backup Server должен быть указан в таблице master..sysservers. Эта запись создается во время установки или обновления и не должна удаляться.
- Если устройства для резервирования расположены на другой машине и загрузка происходит через сеть, на удаленной машине также должен быть установлен сервер Backup Server.

Имена томов

- Тома резервных копий помечаются в соответствии со стандартом ANSI на метки лент. Метка включает номер логического тома и позицию устройства в наборе полос.

- Во время загрузок сервер Backup Server использует метку ленты для проверки правильности порядка, в котором монтируются тома. Это позволяет производить загрузки с меньшего количества устройств, чем использовалось при создании резервных копий.

Примечание. При создании резервных копий и выполнении загрузок через сеть для каждой операции необходимо указывать одинаковое количество устройств с чередованием сегментов/разделов.

Изменение томов резервных копий

- Если сервер Backup Server обнаруживает проблему с монтируемым в данный момент томом, он посылает запрос на замену тома либо клиенту, либо на его консоль оператора. После монтажа другого тома оператор извещает об этом сервер Backup Server посредством выполнения команды sp_volchanged на любом сервере Adaptive Server, который может связываться с Backup Server.

Восстановление системных баз данных

- Пошаговые инструкции по восстановлению системных баз данных из резервных копий см. в книге *Руководство по системному администрированию*.

Зеркальное копирование диска

- В начале загрузки сервер Adaptive Server передает серверу Backup Server имя первичного устройства для каждого устройства логической базы данных и каждого устройства логического журнала. Если режим зеркального копирования для первичного устройства отключен, сервер Adaptive Server вместо этого передает имя вторичного устройства. Если с любым из названных устройств происходит сбой до завершения передачи данных сервером Backup Server, сервер Adaptive Server прекращает загрузку.
- При попытке отключить режим зеркального копирования для любого названного устройства во время выполнения команды load transaction, сервер Adaptive Server выдает сообщение. Пользователь, выполняющий команду disk unmirror, может прекратить загрузку или отложить выполнение команды disk unmirror до завершения загрузки.
- Сервер Backup Server загружает данные в первичное устройство, затем командой load transaction копирует их во вторичное устройство. Команда load transaction требует больше времени для завершения, если какое-либо устройство базы данных зеркалировано.

Стандарты	Уровень соответствия стандарту SQL92: расширение Transact-SQL.
Полномочия	Полномочия на выполнение команды load transaction по умолчанию принадлежат владельцу базы данных и операторам. Эти полномочия нельзя передавать другим пользователям.
См. также	**Команды** disk unmirror, dump database, dump transaction, load database, online database **Системные процедуры** sp_dboption, sp_helpdb, sp_helpdevice, sp_volchanged

lock table

Описание

Явным образом блокирует таблицу на время транзакции.

Синтаксис

lock table *имя_таблицы* in {share | exclusive } mode
[wait [*число_секунд*] | nowait]

Параметры

имя_таблицы
Задает имя таблицы, которую нужно заблокировать.

share | exclusive
Указывает тип применяемой к таблице блокировки (разделяемая или монопольная).

wait число_секунд
Указывает время ожидания в секундах, если блокировка не может быть установлена немедленно. Если параметр *число_секунд* опущен, указывает, что команда lock table должна ожидать, пока не будет предоставлена блокировка.

nowait
Аварийно завершает команду, если блокировка не может быть установлена немедленно.

Примеры

Пример 1. Попытка установить разделяемую блокировку на таблицу titles. Если командой set lock wait был установлен период ожидания на уровне сеанса, то команда lock table ожидает в течение этого периода времени; в противном случае используется период ожидания, установленный на уровне сервера:

```
begin transaction
lock table titles in share mode
```

Пример 2. Попытка установить монопольную блокировку на таблицу authors. Если блокировка не может быть установлена в течение 5 секунд, команда возвращает сообщение с информацией об этом. Последующие команды в транзакции продолжают выполняться, как если бы команды lock table не было:

```
begin transaction
lock table authors in exclusive mode wait 5
```

Пример 3. Если блокировка таблицы не была установлена в течение 5 секунд, то эта процедура проверяет роль пользователя. Если процедуру выполняет пользователь с ролью sa_role, то процедура выдает совет и продолжает выполняться без блокировки таблицы. Если у пользователя нет роли sa_role, то выполняется откат транзакции.

```
create procedure bigbatch
as
begin transaction
lock table titles in share mode wait 5
if @@error = 12207
begin
    /*
    ** Разрешает пользователю SA продолжать процедуру
без блокировки таблицы
    ** Другие пользователи получают сообщение об ошибке
    */
    if (proc_role("sa_role") = 0)
    begin
    print "You cannot run this procedure at
        this time, please try again later"
    rollback transaction
    return 100
    end
else
    begin
    print "Couldn't obtain table lock,
        proceeding with default locking."
    end
end
/* еще операторы SQL */
commit transaction
```

Использование

- Команду lock table можно использовать только в транзакции. Блокировка таблицы удерживается на протяжении транзакции.
- Поведение команды lock table зависит от параметров, задающих период ожидания, которые указаны в команде или активны на уровне сеанса или сервера.
- Если параметр wait или nowait не указан, команда lock table использует период ожидания, указанный или на уровне сеанса, или на уровне сервера. Если период ожидания установлен на уровне сеанса командой set lock wait, используется его значение, в противном случае используется период ожидания, установленный на уровне сервера.
- Если блокировка таблицы не может быть установлена в течение определенного времени (если оно задано), команда lock table возвращает сообщение 12207. Откат транзакции не выполняется. Последующие команды в транзакции продолжают выполняться, как если бы команды lock table не было.

- Команду lock table нельзя использовать для системных или временных таблиц.
- В одной транзакции можно выполнять несколько команд lock table.

Стандарты — Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия — Чтобы использовать команду lock table in share mode, необходимо иметь полномочие select на доступ к таблице. Чтобы использовать команду lock table in exclusive mode, необходимо иметь полномочие delete, insert или update на доступ к таблице.

См. также — **Команды** set

nullif

Описание Поддерживает условные выражения SQL; используется везде, где допускаются значения выражений. Альтернатива выражению case.

Синтаксис nullif(*выражение*, *выражение*)

Параметры

nullif
Сравнивает значения двух выражений. Если первое выражение равно второму, nullif возвращает значение NULL. Если первое выражение не равно второму, nullif возвращает первое выражение.

выражение
Имя столбца, константа, функция, подзапрос или любая комбинация имен столбцов, констант и функций, связанная арифметическими или логическими операторами. Более подробно выражения описаны в разделе “Выражения” на стр. 231.

Примеры **Пример 1.** Выбор столбцов *title* и *type* из таблицы *titles*. Если столбец *type* равен UNDECIDED, то благодаря nullif возвращается значение NULL:

```
select title,
    nullif(type, "UNDECIDED")
from titles
```

Пример 2. Альтернативная запись примера 1:

```
select title,
    case
        when type = "UNDECIDED" then NULL
        else type
    end
from titles
```

Использование

- Выражение nullif является альтернативой выражения case.
- Выражение nullif упрощает стандартные выражения SQL, позволяя указывать условие поиска в виде обычного сравнения вместо конструкции when...then.
- Выражения nullif могут использоваться везде, где допустимы выражения SQL.
- Хотя бы один результат выражения case должен быть не равен NULL. Например, следующий код приводит к сообщению об ошибке:

```
select price, coalesce (NULL, NULL, NULL)
from titles
All result expressions in a CASE expression must not be NULL.
```

- Если запрос выдает несколько типов данных, то тип данных результата выражения case будет определен в соответствии с иерархией типов данных, описанной в разделе "Тип данных смешанных выражений" главы 1, "Системные и пользовательские типы данных". Если Adaptive Server не сможет выполнить неявное преобразование типов (например, для типов char и int), запрос завершается ошибкой.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Выражение nullif по умолчанию могут использовать все пользователи. Для этого не требуется никаких полномочий.

См. также

Команды case, coalesce, select, if...else, Инструкция where

online database

Описание

Помечает базу данных как доступную для общего использования после обычной последовательности загрузки; если требуется, обновляет уже загруженную базу данных до текущей версии Adaptive Server; переводит базу данных в оперативный режим после загрузки журнала транзакций, зарезервированного с использованием параметра for standby_access.

Синтаксис

online database *имя_базы_данных* [for standby_access]

Параметры

имя_базы_данных
Имя базы данных, переводимой в оперативный режим.

for standby_access
Переводит базу данных в оперативный режим (предполагается, что в базе данных нет открытых транзакций).

Примеры

Пример 1. Команда делает базу данных pubs2 доступной для общего пользования после завершения последовательности загрузки:

```
online database pubs2
```

Пример 2. Перевод базы данных inventory_db в оперативный режим. Используется после загрузки inventory_db с резервной копией журнала транзакций, полученной с помощью команды dump tran...with standby_access:

```
online database inventory_db for standby_access
```

Использование

- Команда online database переводит базу данных в оперативный режим для общего пользования после обычной загрузки базы данных или журнала транзакции.
- После выполнения команды load database база данных находится в автономном режиме. Автономный статус устанавливается в системной таблице sysdatabases и сохраняется до окончания выполнения команды online database.
- Команду online database *нельзя* выполнять до того, как будут загружены все журналы транзакций. Последовательность команд должна быть следующей:
  - load database
  - команды load transaction (команд load transaction может быть несколько)
  - online database
- При выполнении команды online database для базы данных, которая уже находится в оперативном режиме, обработка не производится и сообщение об ошибке не выдается.

- Команду online database...for standby_access можно выполнять, только если резервная копия журнала транзакций была сделана с помощью команды dump transaction с параметром with standby_access. Если команда online database...for standby_access выполняется после загрузки журнала транзакций, резервная копия которого была создана без использования команды dump transaction...with standby access, то она аварийно завершается с сообщением об ошибке.
- Определить, в каком режиме находится база данных (в оперативном, оперативном для резервного доступа или автономном), можно с помощью системной процедуры sp_helpdb.

Обновление баз данных до текущей версии сервера

- Команда online database при необходимости выполняет модернизацию загруженной базы данных, резервных копий журналов транзакций, чтобы сделать базу данных совместимой с текущей версией Adaptive Server. По завершении модернизации база данных доступна для общего пользования. Если во время обработки возникают ошибки, база данных остается в автономном режиме.
- Команду online database необходимо выполнять только после последовательности загрузки базы данных или журнала транзакций. Для новых установок или модернизаций она не требуется. При модернизации Adaptive Server до новой версии все базы данных, связанные с сервером, обновляются автоматически.
- Команда online database обновляет пользовательские базы данных только версии 10.0 и выше.
- После обновления базы данных с помощью команды online database необходимо создать резервную копию обновленной базы данных, чтобы иметь копию, согласующуюся с текущей версией Adaptive Server. Резервная копия обновленной базы данных должна быть создана до выполнения команды dump transaction.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду online database могут выполнить только системный администратор, владелец базы данных и пользователь с ролью Operator.

См. также

Команды dump database, dump transaction, load database, load transaction

Системные процедуры sp_helpdb

open

Описание

Открывает курсор для обработки.

Синтаксис

open *имя_курсора*

Параметры

имя_курсора
Имя курсора, который нужно открыть.

Примеры

Команда открывает курсор authors_crsr:

```
open authors_crsr
```

Использование

- Команда open открывает курсор. Курсоры позволяют изменять и удалять строки по одной. Перед использованием операторов fetch, update и delete необходимо открыть курсор. Дополнительную информацию о курсорах см. в книге *Transact-SQL User's Guide*.
- При выполнении команды open будет возвращено сообщение об ошибке, если курсор уже открыт или не был создан оператором declare cursor.
- При открытии курсора выполняется оператор select, определяющий курсор (указанный в операторе declare cursor), и делает результирующий набор курсора доступным для обработки.
- Если курсор открыт в первый раз, указатель устанавливается перед первой строкой результирующего набора курсора.
- Если задан режим связанных транзакций, Adaptive Server неявно начинает транзакцию с оператора open, если нет активных транзакций.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду open по умолчанию могут выполнять все пользователи.

См. также

Команды close, declare cursor, fetch

order by

Описание

Возвращает результаты запроса, отсортированные в указанном порядке по значениям одного или нескольких столбцов.

Синтаксис

[*Начало оператора* select

[order by {[*имя_таблицы*.| *имя_представления*.]*имя_столбца*
| *номер_в_списке_выборки* | *выражение*} [asc | desc]
[,{[*имя_таблицы*.| *имя_представления*.] *имя_столбца*
номер_в_списке_выборки | *выражение*} [asc
|desc]]...]

[*Конец оператора* select

Параметры

order by

Сортирует результаты по столбцам.

asc

Сортирует результаты в порядке возрастания. Если не указано asc или desc, применяется asc.

desc

Сортирует результаты в порядке убывания.

Примеры

Пример 1. Выбор книг, которые стоят дороже $19,99, и их перечисление в алфавитном порядке названий:

```
select title, type, price
from titles
where price > $19.99
order by title
```

```
title
        type         price
-------------------------------------------------------------
         ------------ -------------------------
But Is It User Friendly?
         popular_comp                    22.95
Computer Phobic and Non-Phobic Individuals: Behavior Variations
          psychology                     21.59
Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean
         trad_cook                       20.95
Secrets of Silicon Valley
         popular_comp                   20.00
```

Пример 2. Перечисление книг из таблицы titles, отсортированных по типу (столбец type) в порядке, обратном алфавитному, и вычисление средней цены и аванса для каждого типа:

```
select type, price, advance
from titles
order by type desc
compute avg(price), avg(advance) by type
```

Пример 3. Перечисление идентификаторов книг из таблицы titles, а также авансов, деленных на суммарные продажи, в последовательности от наименьшего вычисленного значения к наибольшему:

```
select title_id, advance/total_sales
from titles
order by advance/total_sales
```

```
 title_id
 -------- -----------------------
 MC3026                    NULL
 PC9999                    NULL
 MC2222                    0.00
 TC4203                    0.26
 PS3333                    0.49
 BU2075                    0.54
 MC3021                    0.67
 PC1035                    0.80
 PS2091                    1.11
 PS7777                    1.20
 BU1032                    1.22
 BU7832                    1.22
 BU1111                    1.29
 PC8888                    1.95
 TC7777                    1.95
 PS1372                   18.67
 TC3218                   18.67
 PS2106                   54.05
```

Пример 4. Печать списка названий книг (title) и типов (type). Выходные данные сортируются в порядке увеличения значения типа, а столбцы в выходных данных переименовываются:

```
select title as BookName, type as Type
from titles
order by Type
```

Использование

- order by возвращает результаты запроса, отсортированные по одному или нескольким столбцам. order by – это часть команды select.
- В Transact-SQL в инструкции order by можно указывать элементы, которых нет в списке выборки. Сортировать можно по заголовку столбца, имени столбца, выражению, псевдониму (если он указан в списке выборки) или по позиции элемента в списке выборки (*номер_ в_списке_выборки*).

- Если сортировка производится по *номеру_в_списке_выборки*, столбцы, указанные в инструкции order by, должны быть включены в список выборки, при этом список выборки не должен быть * (звездочкой).
- Инструкция order by позволяет отобразить результаты запроса в порядке, имеющем смысл для пользователя. Без инструкции order by нельзя управлять последовательностью, в которой Adaptive Server возвращает результаты запроса.

Ограничения

- Максимально допустимое число столбцов в инструкции order by – 31.
- В инструкции order by нельзя указывать столбцы типов данных text или image.
- Инструкция order by не может содержаться в определении подзапросов и представлений (а также в инструкции compute и вместе с ключевым словом into). И наоборот, инструкция order by не может содержать подзапрос.
- Набор результатов серверного курсора или языкового курсора нельзя обновлять, если оператор select, определяющий этот курсор, содержит инструкцию order by. Дополнительную информацию об ограничениях на обновляемые курсоры см. в книге *Transact-SQL User's Guide*.
- Если указана инструкция compute by, то необходимо также указать инструкцию order by. После инструкции compute by должны быть перечислены те же выражения, что и после инструкции order by, или их подмножество. Они должны быть в том же порядке, начинаться с того же выражения, и ни одно выражение не должно быть пропущено. Например, если инструкция order by выглядит следующим образом:

  ```
  order by a, b, c
  ```

  то инструкция compute by может быть любой из следующих:

  ```
  compute by a, b, c
  compute by a, b
  compute by a
  ```

  Ключевое слово compute может использоваться без by для вычисления общих итогов. В этом случае инструкция order by необязательна.

Порядок сортировки

- В отсортированном наборе, построенном по инструкции order by, значения NULL предшествуют всем остальным.

- Порядок сортировки (упорядочения) на сервере Adaptive Server определяет, как сортируются данные. Существуют следующие порядки сортировки: двоичный, словарный, не различающий регистр, не различающий регистр и с предпочтением, не различающий регистр и диакритические знаки. Также можно указать порядки сортировки для определенных языков.

Таблица 7-29. Последствия выбора порядка сортировки

Порядок сортировки Adaptive Server	**Влияние на результаты инструкции *order by***
Двоичный порядок	Сортирует все данные по числовым байтовым эквивалентам символов. При двоичном порядке сортировки все прописные буквы идут перед строчными. Двоичная сортировка – это единственный способ сортировки для многобайтовых наборов символов.
Словарный порядок	При этом порядке сортировки прописные буквы идут перед соответствующими им строчными буквами (то есть он зависит от регистра). Словарный порядок распознает различные виды букв с диакритическими знаками и ставит их после тех же букв без диакритического знака.
Словарный порядок, не различающий регистр	Сортирует данные в словарном порядке, но не различает регистр символов. Прописные буквы эквивалентны их строчным аналогам и сортируются, как описано ниже в разделе “Правила сортировки”.
Словарный порядок, не различающий регистр, и с предпочтением.	При этом порядке заглавная буква идет перед строчным аналогом. Однако при сравнении (например в инструкции where) регистр букв не учитывается.
Словарный порядок, не различающий регистр и диакритические знаки.	Сортирует данные в словарном порядке, не различая регистр; при этом порядке буквы с диакритическими знаками считаются эквивалентными их аналогам без диакритического знака. В результате в отсортированном наборе буквы с диакритическими знаками не будут упорядочены относительно аналогичных букв без диакритических знаков.

- О том, какой порядок сортировки установлен на сервере Adaptive Server, можно узнать из системной процедуры sp_helpsort.

Правила сортировки

- Когда две строки эквивалентны относительно порядка сортировки, применяются следующие правила:
  - Сравниваются значения в столбцах, указанных в инструкции order by.
  - Если две строки имеют одинаковые значения в этих столбцах, производится побайтное сравнение двоичных значений, построенных по каждой строке целиком. При сравнении используется порядок, в котором столбцы физически хранятся, а не тот порядок, в котором они указаны в запросе или команде create table,

создавшей таблицу. Порядок физического хранения можно кратко описать следующим образом: сначала хранятся столбцы фиксированной длины (в порядке, указанном при создании таблицы), а затем – столбцы переменной длины (также в порядке, указанном при создании таблицы).

- Если строки равны, сравниваются идентификаторы строк.

  Рассмотрим следующую таблицу:

```
create table sortdemo (lname varchar(20),
                       init char(1) not null)
```

  и такие данные:

```
lname      init
---------- ----
Smith      B
SMITH      C
smith      A
```

  Результаты сортировки по столбцу *lname* будут следующими:

```
lname      init
---------- ----
smith      A
Smith      B
SMITH      C
```

  Поскольку данные фиксированной длины типа char (столбец init) физически хранятся первыми, order by сортирует эти строки на основе двоичных значений “Asmith”, “BSmith” и “CSMITH”.

  Однако если бы столбец init имел тип varchar, то первым хранился бы столбец *lname*, а только затем – столбец init. В этом случае сравнение происходило бы по двоичным значениям “SMITHC”, “SmithB” и “smithA” и строки были бы возвращены именно в этом порядке.

Сканирование по убыванию

- Использование ключевого слова desc в инструкции order by позволяет оптимизатору запросов выбрать стратегию, которая избавляет от необходимости создания рабочей таблицы и выполнения сортировки для возвращения результатов в порядке убывания. В результате такой оптимизации цепочка страниц индекса просматривается в обратном порядке, следуя указателям на предыдущие страницы на каждой странице индекса.

Чтобы использовать эту оптимизацию, порядок столбцов в инструкции order by должен быть таким же, как и в индексе. При этом столбцы в инструкции order by могут быть подмножеством ключей индекса, но это подмножество должно быть префиксным, то есть содержать только первые ключи. Оптимизацию, основанную на сканировании по убыванию, нельзя использовать, если множество столбцов, указанных в order by, включает в себя множество индексных ключей.

Если запрос содержит соединение, все таблицы могут быть просканированы в порядке убывания ключа, если только соблюдаются требования относительно префиксного подмножества ключей. Оптимизация, основанная на сканировании в порядке убывания ключей, может применяться не для всех, а только для некоторых таблиц в соединении (а остальные таблицы будут сканироваться по возрастанию).

- Если другие пользовательские процессы сканируют по возрастанию при обновлении или удалении, то сканирование в порядке убывания может вызвать взаимоблокировки. Взаимоблокировки могут также возникнуть при расщеплениях страниц и освобождении страниц из-под индекса (page shrink). Получить сведения о взаимоблокировках на сервере можно с помощью системной процедуры sp_sysmon, а записать информацию о взаимоблокировках в журнал ошибок – с помощью установки параметра конфигурации print deadlock information.
- Если приложения требуют, чтобы результаты возвращались в порядке убывания, но сканирование по убыванию вызывает взаимоблокировки, возможны следующие решения:
  - Можно провести сканирование по убыванию с нулевым уровнем изоляции транзакций. Дополнительную информацию о влиянии уровня изоляции 0 на операции чтения см. в книге *Руководство по настройке производительности*.
  - Возможность сканирования по убыванию можно отключить с помощью параметра конфигурации allow backward scans, чтобы все запросы, в которых указано ключевое слово desc, сканировали таблицу в порядке возрастания, и отсортировать набор результатов по убыванию. Дополнительную информацию см. в книге *Руководство по системному администрированию*.

    - Проблемные операции сканирования по убыванию можно разбить на два шага: выбор нужных строк во временную таблицу в порядке возрастания и выбор этих строк из временной таблицы в порядке убывания.

- Если сканирование по убыванию использует кластерный индекс, который содержит страницы переполнения из-за наличия одинаковых значений ключа, порядок возвращаемых им результатов может не быть в точности обратным порядку результатов, возвращаемых сканированием по возрастанию. Указанные значения ключей возвращаются в нужном порядке, но порядок строк с одинаковыми значениями ключей на страницах переполнения может отличаться. Механизм хранения страниц переполнения в кластерных индексах объясняется в книге *Руководство по настройке производительности*.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Возможность указывать новые заголовки столбцов в инструкции order by команды select, содержащей оператор union, является расширением Transact-SQL.

См. также

Команды compute, declare, group by и having, select, Инструкция where

Системные процедуры sp_configure, sp_helpsort, sp_lock, sp_sysmon

prepare transaction

Описание	Используется DB-Library в приложениях с двухфазной фиксацией транзакций; позволяет определить, готов ли сервер зафиксировать транзакцию.
Синтаксис	prepare tran[saction]
Использование	• Дополнительную информацию см. в книге *Open Client DB-Library Reference Manual*.
Стандарты	Уровень соответствия стандарту SQL92: расширение Transact-SQL.
См. также	**Команды** begin transaction, begin transaction, rollback, save transaction

print

Описание

Выводит пользовательское сообщение на экран.

Синтаксис

print
{*строка_формата* | *@локальная_переменная* |
@@глобальная_переменная}
[, *список_аргументов*]

Параметры

строка_формата

Переменная или строка символов. Максимальная длина *строки_формата* – 1023 байта.

Строки формата могут содержать до 20 уникальных указателей места подстановки в любом порядке. Когда текст сообщения будет отправлен клиенту, эти указатели места подстановки будут заменены форматированным содержимым аргументов, которые идут за *строкой_формата*.

Чтобы иметь возможность изменить порядок следования аргументов при переводе строк формата на язык с другой грамматической структурой, указатели места подстановки нумеруются. Указатель места подстановки для аргумента выглядит так: " %*nn* !"– знак процента (%), за ним целое число от 1 до 20 и восклицательный знак (!). Целое число представляет номер аргумента в строке в списке аргументов. Например, "%1!" – это первый аргумент в первоначальной версии, "%2!" – второй аргумент и т.д.

Обозначение позиции аргумента таким способом позволяет правильно перевести текст, даже если порядок аргументов в целевом языке отличается.

Возьмем такое сообщение на английском:

```
%1! is not allowed in %2!.
```

Немецкая версия этого сообщения выглядит так:

```
%1! ist in %2! nicht zulassig.
```

@локальная_переменная

Должна иметь тип char, nchar, varchar или nvarchar и должна быть объявлена в пакете или процедуре, в которой используется.

@@глобальная_переменная

Должна иметь тип char или varchar или допускать автоматическое преобразование к этим типам (например *@@version*). В данный момент *@@version* – это единственная глобальная переменная символьного типа.

список_аргументов

Может быть последовательностью переменных или констант, разделенных запятыми. *список_аргументов* необязателен, если не указана строка формата, содержащая указатели места подстановки вида "*%nn* !". Если строка формата содержит указатели места подстановки, то *список_аргументов* должен содержать количество аргументов, соответствующее максимальному номеру указателя места подстановки. Аргумент может быть любого типа данных, кроме text или image; перед включением в конечное сообщение он преобразуется в символьный тип.

Примеры

Пример 1. Печать строки "Berkeley author", если автор из таблицы authors проживает в области с почтовым индексом (столбец postacode) 94705:

```
if exists (select postalcode from authors
where postalcode = '94705')
print "Berkeley author"
```

Пример 2. Объявление переменной, присвоение ей значения и его вывод на экран:

```
declare @msg char(50)
select @msg = "What's up, doc?"
print @msg
```

```
What's up, doc?
```

Пример 3. Использование переменных и указателей места подстановки в сообщениях:

```
declare @tabname varchar(30)
select @tabname = "titles"

declare @username varchar(30)
select @username = "ezekiel"

print "The table '%1!' is not owned by the user '%2!'.",
@tabname, @username
```

```
The table 'titles' is not owned
by the user 'ezekiel'.
```

Использование

- Максимальная длина выводимой *строки_формата* (со всеми аргументами после подстановки) равна 1024 байтам.
- Если строка формата содержит указатель места подстановки с номером *n*, то в этой же строке должны присутствовать указатели места подстановки с номерами от 1 до *n*- 1, хотя и не обязательно в порядке номеров. Например, недопустимо, чтобы в строке формата

были указатели места подстановки 1 и 3, но не было указателя места подстановки 2. Если в строке формата пропущен номер, то при выполнении команды print будет выдано сообщение об ошибке.

- *список_аргументов* должен содержать аргумент для каждого указателя места подстановки в *строке_формата*, иначе выполнение транзакции будет прекращено. Аргументов может быть больше, чем указателей места подстановки.
- Чтобы вставить знак процента в сообщение об ошибке, в *строке_формата* нужно указать два знака процента (“%%”). Если *строка_формата* содержит один знак процента (“%”), который не используется как указатель места подстановки, то будет возвращено сообщение об ошибке.
- Если аргумент принимает значение NULL, то он преобразуется в символьную строку нулевой длины. Чтобы избежать вывода строк нулевой длины, используется функция isnull. Например, если аргумент *@arg* имеет значение NULL, то следующий оператор выдает сообщение `"I think we have nothing here."`:

```
declare @arg varchar(30)
select @arg = isnull(col1, "nothing") from
table_a where ...
print "I think we have %1! here", @arg
```

- Пользовательские сообщения можно добавлять в системную таблицу sysusermessages, чтобы использовать их в любом приложении. Для добавления сообщений в таблицу sysusermessages используется системная процедура sp_addmessage, а для извлечения сообщений для команд print и raiserror – процедура sp_getmessage.
- Чтобы вывести определенное пользователем сообщение об ошибке и сохранить номер ошибки в переменной *@@error*, нужно использовать команду raiserror, а не print.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду print по умолчанию могут выполнять все пользователи. Для этого не требуется никаких полномочий.

См. также

Команды declare, raiserror

Системные процедуры sp_addmessage, sp_getmessage

quiesce database

Описание

Приостанавливает и возобновляет операции обновления в перечисленных базах данных.

Синтаксис

quiesce database *имя_тега* hold *имя_базы_данных* [, *имя_базы_данных*] ... [for external dump]

или:

quiesce database *имя_тега* release

Параметры

имя_тега

Имя, заданное пользователем и указывающее список баз данных, обновления в которых нужно приостановить (hold) или возобновить (release). *имя_тега* должно соответствовать правилам для идентификаторов.

база_данных

Имя базы данных.

for external dump

Указывает, что, пока обновление в перечисленных базах данных приостановлено, средства, не являющиеся частью Adaptive Server, будут физически копировать все соответствующие устройства баз данных. Операция копирования служит заменой комбинации команд dump database и load database.

Примеры

Пример 1. Приостановка обновлений в базах данных salesdb и ordersdb:

```
quiesce database report_dbs hold salesdb, ordersdb
```

Пример 2. Возобновление обновлений в базах данных, помеченных тегом report_dbs:

```
quiesce database report_dbs release
```

Пример 3. Приостановка обновлений в базе данных pubs2 с целью создать внешнюю копию этой базы данных:

```
quiesce database pubs_tag hold pubs2 for external dump
```

Использование

- Команда quiesce database с ключевым словом hold приостанавливает все обновления в указанной базе данных. Транзакции не могут обновлять данные в приостановленных базах данных, а фоновые задачи, например, процесс контрольной точки и процесс Housekeeper, пропускают все приостановленные базы данных.
- Команда quiesce database с ключевым словом release позволяет возобновить обновления баз данных, которые раньше были приостановлены.

- Команда quiesce database с инструкцией for external dump указывает на намерение создать внешнюю копию базы данных. Эта инструкция заменяет комбинацию команд резервного копирования и загрузки базы данных.
- Команды quiesce database hold и release необязательно выполнять в одном и том же пользовательском сеансе.
- Если базы данных, указанные в команде quiesce database hold, содержат распределенные или межбазовые транзакции в подготовленном состоянии, Adaptive Server ожидает завершения этих транзакций пять секунд. Если за это время транзакции не закончатся, quiesce database hold завершается с ошибкой.
- Если над какой-либо из баз данных, указанной в команде quiesce database hold, выполняется команда dump database или dump transaction, эта база данных приостанавливается только по окончании выполнения соответствующей команды dump.
- Если над базой данных, в которой приостановлены обновления, попытаться выполнить команду dump database или dump transaction, эта команда будет заблокирована до тех пор, пока база данных не будет освобождена командой quiesce database release.
- Максимальное количество баз данных в одной команде quiesce database hold – 8. Если требуется приостановить обновление большего числа баз данных, нужно выполнить команду quiesce database hold несколько раз.

Полномочия

Команду quiesce database по умолчанию могут выполнять системные администраторы.

См. также

Команды dump database, dump transaction

Системные процедуры sp_helpdb, sp_who

raiserror

Описание

Выводит на экран пользовательское сообщение об ошибке и устанавливает системный флаг для регистрации возникновения ошибки.

Синтаксис

raiserror *номер_ошибки*
[{*строка_формата* | *@локальная_переменная*}]
[, *список_аргументов*]
[with errordata *ограниченный_список_выборки*]

Параметры

номер_ошибки

Локальная переменная или целое число больше 17000. Если *номер_ошибки* находится в диапазоне от 17000 до 19999, а *строка_формата* отсутствует или пуста (“”), Adaptive Server ищет текст сообщения об ошибке в таблице sysmessages базы данных master. Эти сообщения об ошибках используются в основном системными процедурами.

Если *номер_ошибки* больше или равен 20000, а *строка_формата* отсутствует или пуста, команда raiserror ищет текст сообщения в таблице sysusermessages в базе данных, из которой поступил запрос или была вызвана хранимая процедура. Adaptive Server пытается отыскать сообщения в sysmessages или sysusermessages на языке, указанном в параметре *@@langid*.

строка_формата

Строка символов с максимальной длиной 1024 байта. При желании можно объявить *строку_формата* в локальной переменной и использовать ее с raiserror (см. *@локальная_переменная*).

Команда raiserror распознает указатели места подстановки в символьной строке, которая должна быть выведена. Строки формата могут содержать до 20 уникальных указателей места подстановки в любом порядке. Когда текст сообщения будет отправлен клиенту, эти указатели места подстановки будут заменены форматированным содержимым аргументов, которые идут за *строкой_формата*.

Чтобы иметь возможность изменить порядок следования аргументов при переводе строки формата на язык с другой грамматической структурой, указатели места подстановки нумеруются. Указатель места подстановки для аргумента выглядит так: %nn!”– знак процента (%), за ним целое число от 1 до 20 и восклицательный знак (!). Целое число представляет номер аргумента в строке в списке аргументов: “%1!” – первый аргумент в исходной версии, “%2!” – второй аргумент и т. д.

Обозначение позиции аргумента таким способом позволяет корректно перевести текст, даже если порядок перечисления аргументов в целевом языке отличается.

Возьмем такое сообщение на английском:

```
%1! is not allowed in %2!.
```

Немецкая версия этого сообщения такова:

```
%1! ist in %2! nicht zulassig.
```

@локальная_переменная
Локальная переменная, содержащая *строку_формата*. Она должна быть типа char или varchar и должна быть объявлена в пакете или процедуре, где она используется.

список_аргументов
Последовательность переменных или констант, разделенных запятыми. *Список_аргументов* необязателен, если не указана строка формата, содержащая указатели мест подстановки вида "*%nn* !". Аргумент может быть любого типа данных, кроме text и image; он преобразуется в тип char перед включением в конечное сообщение.

Если аргумент равен NULL, Adaptive Server преобразует его в символьную строку нулевой длины типа char.

with errordata
Предоставляет расширенные данные об ошибках для программ Client-Library™.

ограниченный_список_выборки
Состоит из одного или нескольких следующих элементов:

- Символа "*", представляющего все столбцы в том порядке, в котором они указаны в команде create table.
- Списка имен столбцов в том порядке, в котором они должны появляться в результате. Чтобы выбрать существующий столбец IDENTITY, можно вместо фактического имени этого столбца указать ключевое слово syb_identity, при необходимости уточненное именем таблицы.
- Инструкции, добавляющей столбец IDENTITY в результирующую таблицу:

  ```
  имя_столбца = identity(точность)
  ```

- Псевдонима, который будет использоваться вместо заголовка столбца по умолчанию (которым является имя столбца), в одной из следующих форм:

  ```
  заголовок_столбца = имя_столбца
  имя_столбца заголовок_столбца
  имя_столбца as заголовок_столбца
  ```

Заголовок столбца может быть заключен в кавычки для любой из этих форм. Заголовок должен быть заключен в кавычки, если он не является допустимым идентификатором (то есть является зарезервированным словом, начинается со специального символа либо содержит пробелы или знаки пунктуации).

- Выражения (имени столбца, константы, функции или любой комбинации имен столбцов, констант и функций, связанных арифметическими или логическими операторами, а также подзапроса).
- Встроенной или агрегатной функции.
- Любой комбинации вышеперечисленных элементов.

ограниченный_список_выборки также может присваивать значение переменной с помощью следующего синтаксиса:

```
@переменная =  выражение)
[, @переменная =  выражение...)
```

На *ограниченный_список_выборки* накладываются следующие ограничения:

- В *ограниченном_списке_выборки* нельзя совмещать присваивание значения переменной с другими параметрами.
- В *ограниченном_списке_выборки* нельзя использовать инструкции from, where и другие инструкции команды select.
- В *ограниченном_списке_выборки* нельзя указывать "*", чтобы выбрать все столбцы.

Дополнительную информацию см. в книге *Transact-SQL User's Guide*.

Примеры

Пример 1. Эта хранимая процедура возвращает ошибку, если не находит таблицу, указанную в параметре *@tabname*:

```
create procedure showtable_sp @tabname varchar(18)
as
if not exists (select name from sysobjects
    where name = @tabname)
    begin
        raiserror 99999 "Table %1! not found.",
        @tabname
    end
else
    begin
```

```
        select sysobjects.name, type, crdate, indid
        from sysindexes, sysobjects
        where sysobjects.name = @tabname
        and sysobjects.id = sysindexes.id
    end
```

Пример 2. Этот пример добавляет сообщение в таблицу sysusermessages, после чего тестирует это сообщение с помощью команды raiserror, предоставляя аргументы для подстановки:

```
sp_addmessage 25001,
"There is already a remote user named '%1!'
for remote server '%2!'."

raiserror 25001, jane, myserver
```

Пример 3. В этом примере используется параметр with errordata для возврата расширенных данных об ошибке *column* и *server* приложению клиента, чтобы показать, какой столбец был вовлечен в ошибку и какой сервер при этом использовался:

```
raiserror 20100 "Login must be at least 5
    characters long" with errordata "column" =
    "login", "server" = @@servername
```

Использование

- Пользовательские сообщения могут вводиться в самой команде raiserror, как в примерах 1 и 3, или добавляться в системную таблицу sysusermessages для использования в других приложениях, как в примере 2. Для добавления сообщений в таблицу sysusermessages используется процедура sp_addmessage, а для извлечения сообщений для команд print и raiserror – процедура sp_getmessage.
- Номера ошибок для пользовательских сообщений об ошибках должны быть больше 20000. Максимальное значение – 2 147 483 647 (2^{31} -1).
- Уровень серьезности для всех пользовательских сообщений об ошибках – 16. Этот уровень означает, что пользователь совершил нефатальную ошибку.
- Максимальная длина выводимой *строки_формата* (со всеми аргументами после подстановки) равна 1024 байтам.
- Если строка формата содержит указатель места подстановки с номером *n*, то в этой же строке должны присутствовать указатели места подстановки с номерами от 1 до *n*- 1, хотя и не обязательно в порядке номеров. Например, недопустимо, чтобы в строке формата были указатели мест подстановки с номерами 1 и 3, но не было указателя 2.

Если в строке формата пропущен указатель места подстановки с тем или иным номером, то при выполнении команды raiserror будет выдано сообщение об ошибке.

- Если аргументов меньше, чем указателей мест подстановки в *строке_формата*, то будет выдано сообщение об ошибке, и транзакция будет прервана. Аргументов может быть больше, чем указателей мест подстановки в *строке_формата*.
- Чтобы вставить литеральный знак процента в сообщение об ошибке, необходимо указать два знака процента (“%%”) в строке *строка_формата*. Если *строка_формата* содержит один знак процента (“%”), который не используется как указатель места подстановки, то будет возвращено сообщение об ошибке.
- Если аргумент равен NULL, то он будет преобразован в символьную строку нулевой длины типа char. Чтобы избежать вывода строк нулевой длины, используется функция isnull.
- При выполнении команды raiserror номер ошибки помещается в глобальную переменную *@@error*. В этой переменной хранится номер последней ошибки, сгенерированной системой.
- Одно из отличий команды raiserror от команды print состоит в том, что первая позволяет сохранить номер ошибки в переменной *@@error*.
- Чтобы включить *список_аргументов* в команду raiserror, поставьте запятую после *номера_ошибки* или *строки_формата* перед первым аргументом. Чтобы включить расширенные данные об ошибке, нужно отделить первое *расширенное_значение* от переменных *номер_ошибки*, *строка_формата* или *список_аргументов* пробелом (а не запятой).

Стандарты Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия Команду raiserror по умолчанию могут выполнять все пользователи. Для этого не требуется никаких полномочий.

См. также

Команды declare, print

Системные процедуры sp_addmessage, sp_getmessage

readtext

Описание

Читает указанное количество байтов или символов значения типа text или image, начиная с указанного смещения.

Синтаксис

```
readtext [[база_данных.]владелец.]имя_таблицы.имя_столбца
        указатель_на_данные_типа_text размер смещения
        [holdlock | noholdlock] [readpast]
        [using {bytes | chars | characters}]
        [at isolation {
            [ read uncommitted | 0 ] |
            [ read committed | 1 ] |
            [ repeatable read | 2 ]|
            [ serializable | 3 ] } ]
```

Параметры

имя_таблицы.имя_столбца

Имя столбца с типом text или image. Имя таблицы указывать обязательно. Если таблица находится в другой базе данных, также нужно указать имя базы данных, а если в базе данных существует несколько таблиц с таким именем, то нужно также указать имя владельца. По умолчанию *владелец* – это текущий пользователь, а *база_данных* – текущая база данных.

указатель_на_данные_типа_text

Значение varbinary(16), которое хранит указатель на данные типа text или image. Для определения этого значения используется функция textptr (см. пример 1). Данные типа text и image хранятся отдельно от остальных столбцов таблицы (в отдельном наборе связанных страниц).Указатель на их действительное местоположение хранится вместе с данными; функция textptr возвращает этот указатель.

смещение

Указывает количество байтов или символов, которые должны быть пропущены перед считыванием данных text или image.

размер

Указывает количество считываемых байтов или символов.

holdlock

Блокирует текстовое значение для операций чтения до конца транзакции. Другие пользователи могут читать значение, но не могут его изменить.

noholdlock

Запрещает серверу удерживать любые блокировки, установленные во время выполнения этого оператора, независимо от текущего уровня изоляции транзакций. В одном запросе нельзя указывать оба параметра – holdlock и noholdlock.

readpast
: Указывает, что команда readtext должна пропустить строки с монопольными блокировками, не ожидая снятия этих блокировок и не выдавая сообщение.

using
: Указывает, каким образом команда readtext интерпретирует параметры *смещение* и *размер* – как количество байтов (bytes) или как количество символов textptr (chars или characters, они являются синонимами). Этот параметр не учитывается, когда команда используется с однобайтовым набором символов или со значениями типа image (команда readtext считывает значения типа image побайтно). Если параметр using отсутствует, readtext интерпретирует аргументы *размер* и *смещение* как байты.

at isolation
: Указывает уровень изоляции запроса (0, 1 или 3). Если опустить эту инструкцию, запрос использует уровень изоляции сеанса, в котором он выполняется (уровень изоляции по умолчанию равен 1). Если ключевое слово holdlock указано в запросе, где также содержится at isolation read uncommitted, то сервер Adaptive Server выдает предупреждение и игнорирует инструкцию at isolation. Для других уровней изоляции holdlock имеет более высокий приоритет, чем at isolation.

read uncommitted
: Задает уровень изоляции 0 для запроса. Вместо read uncommitted можно указать 0 с инструкцией at isolation.

read committed
: Задает уровень изоляции 1 для запроса. Вместо read committed можно указать 1 с инструкцией at isolation.

repeatable read
: Задает уровень изоляции 2 для запроса. Вместо serializable можно указать 2 с инструкцией at isolation.

serializable
: Задает уровень изоляции 3 для запроса. Вместо serializable можно указать 3 с инструкцией at isolation.

Примеры

Пример 1. Выбор символов со второго по шестой в столбце copy:

```
declare @val varbinary(16)
select @val = textptr(copy) from blurbs
where au_id = "648-92-1872"
readtext blurbs.copy @val 1 5 using chars
```

Пример 2.

```
declare @val varbinary(16)
select @val = textptr(copy) from blurbs readpast
where au_id = "648-92-1872"
readtext blurbs.copy @val 1 5 readpast using chars
```

Использование

- Функция textptr возвращает 16-байтную двоичную строку (указатель на данные типа text) в столбец text или image указанной строки или в столбец text или image последней строки, возвращенной запросом, если запрос возвращает более одной строки. Лучше всего объявить локальную переменную для хранения указателя на данные типа text, а потом использовать эту переменную в readtext.
- Значение глобальной переменной *@@textsize*, которое ограничивает количество возвращаемых байтов данных, замещает размер, указанный для readtext, если он меньше размера, хранимого в переменной. Изменить значение переменной *@@textsize* можно с помощью команды set textsize.
- Если смещение и размер заданы в байтах, Adaptive Server может найти неполные символы в начале или в конце возвращаемых данных text. Если в этой ситуации включено преобразование набора символов, сервер заменяет каждый неполный символ на знак вопроса (?) перед возвращением текста клиенту.
- Adaptive Server должен определить количество байтов, отправляемых клиенту в ответ на команду readtext. Если *смещение* и *размер* заданы в байтах, определить количество байтов в возвращаемом тексте несложно. Если смещение и размер заданы в символах, сервер должен вычислить количество байтов, возвращаемых клиенту. Это дополнительное вычисление может снизить производительность. Параметр using characters полезен только в случае, если сервер Adaptive Server использует многобайтовый набор символов. Он гарантирует, что readtext не будет возвращать неполные символы.
- Нельзя использовать readtext в отношении столбцов с типом text и image в представлениях.
- Если после перехода на многобайтовый набор символов не была выполнена проверка dbcc fix_text, то при попытке использовать команду readtext для значений типа text эта команда даст сбой и будет выведено сообщение об ошибке, в котором рекомендуется выполнить dbcc fix_text для таблицы.

Использование параметра *readpast*

- readpast применяется только к таблицам с блокировкой только данных. readpast игнорируется, если указывается для таблицы с блокировкой всех страниц.
- Параметр readpast несовместим с параметром holdlock. Если в команде указаны оба параметра, выдается ошибка и выполнение команды прерывается.
- Если в readtext указано at isolation read uncommitted, readpast выдает предупреждение, но не прерывает выполнение команды.
- Если уровень изоляции оператора — 3, readpast выдает ошибку и прерывает выполнение команды.
- Если уровень изоляции сеанса – 3, readpast игнорируется без предупреждения.
- Если уровень изоляции сеанса – 0, readpast выдает предупреждение, но не прерывает выполнение команды.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду readtext могут выполнять только пользователи, обладающие полномочием select для таблицы. Полномочие readtext передается вместе с полномочием select.

См. также

Команды set, writetext

Системные процедуры Раздел Типы данных text и image

reconfigure

Описание	Команда reconfigure в настоящий момент не работает. Она оставлена лишь для того, чтобы существующие сценарии можно было выполнять без изменения. В предыдущих версиях для того, чтобы новые значения параметров конфигурации вступили в силу, требовалось после выполнения системной процедуры sp_configure запустить команду reconfigure.
Синтаксис	reconfigure
Использование	**Примечание.** Необходимо переписать сценарии, содержащие команду reconfigure. Хотя команда reconfigure и включена в эту версию, ее поддержка в будущих версиях не гарантирована.
Стандарты	Уровень соответствия стандарту SQL92: расширение Transact-SQL.
Полномочия	Команду reconfigure по умолчанию могут выполнять системные администраторы. Это полномочие нельзя передавать.
См. также	**Системные процедуры** sp_configure

remove java

Описание

Удаляет из базы данных один или несколько классов Java-SQL, пакетов или JAR-архивов.

Применяется, когда Java-классы установлены в базу данных. Дополнительную информацию см. в книге *Java in Adaptive Server Enterprise*.

Синтаксис

```
remove java
        класс имя_класса [, имя_класса]...
            | package имя_пакета [, имя_пакета]...
            | jar имя_jar [,  имя_jar]...[retain classes]
```

Параметры

class имя_класса

Имя одного или нескольких Java-классов, которые нужно удалить из базы данных. Классы должны быть установлены в текущей базе данных.

package имя_пакета

Имя одного и нескольких Java-пакетов, которые будут удалены. Эти пакеты должны быть установлены в текущей базе данных.

jar имя_jar

SQL-идентификатор или значение символьной строки длиной до 30 байтов, содержащие корректный SQL-идентификатор.

Каждое значение *имя_jar* должно соответствовать имени хранимого JAR-архива в текущей базе данных.

retain classes

Указывает, что перечисленные JAR больше не хранятся в базе данных, а хранимые классы не связаны с JAR.

Использование

- Если оператор remove java содержится в хранимой процедуре, под текущей базой данных подразумевается база данных, которая была текущей во время создания процедуры, а не та база данных, которая является текущей в момент вызова процедуры.

  Если оператор remove java не содержится в хранимой процедуре, то текущая база – это база данных, которая является текущей при выполнении оператора remove.

- Если указаны инструкции class или package и любой из удаленных классов имеет связанный с ним JAR-архив, генерируется исключение.

- Если любая хранимая процедура, таблица или представление содержит ссылку на удаленный класс в виде типа данных столбца, переменной или параметра, генерируется исключение.

- Все удаленные классы обрабатываются так.
  - Удаляются из текущей базы данных.
  - Выгружаются из виртуальной машины Java (Java VM) текущего подключения. Удаленные классы не выгружаются из Java VM других подключений.
- Если во время выполнения remove java генерируется какое-то исключение, все действия remove java отменяются.
- Нельзя удалить класс Java-SQL, если на него есть прямая ссылка в функции или хранимой процедуре SQLJ.
- Чтобы удалить класс Java-SQL из базы данных, необходимо сделать следующее.
  a Удалить все функции и хранимые процедуры SQLJ, которые напрямую ссылаются на класс, командой drop procedure и/или drop function.
  b Удалить класс Java-SQL из базы данных с помощью remove java.

Блокировки

- При использовании remove java монопольно блокируется таблица sysxtypes.
- Если указана инструкция jar, монопольно блокируется таблица sysjars.

Полномочия

Команду remove java могут выполнять только системный администратор или владелец базы данных.

См. также

Системные процедуры sp_helpjava

Системные таблицы sysjars, sysxtypes

Утилиты extractjava, installjava

reorg

Описание

Освобождает неиспользуемое пространство на страницах, устраняет перемещения строк или переписывает все строки таблицы на новые страницы в зависимости от указанного параметра.

Синтаксис

reorg reclaim_space *имя_таблицы* [*имя_индекса*]
[with {resume, time = *количество_минут*}]

reorg forwarded_rows *имя_таблицы*
[with {resume, time = *количество_минут*}]

reorg compact *имя_таблицы*
[with {resume, time = *количество_минут*}]

reorg rebuild *имя_таблицы* [*имя_индекса*]

Параметры

reclaim_space
Делает неиспользуемое пространство, которое осталось после операций удаления и обновления, доступным для дальнейшего использования. Если в результате зафиксированных операций удаления или операций обновления, сокращающих длину строки, на странице данных в таблице остается неиспользованное пространство, reorg reclaim_space последовательно переписывает существующие строки, "сдвигая" все неиспользованное пространство в конец страницы. Если на странице нет строк, страница освобождается.

имя_таблицы
Указывает имя таблицы, которая должна быть реорганизована. Если указано *имя_индекса*, реорганизуется только этот индекс.

имя_индекса
Указывает имя индекса, который будет реорганизован.

with resume
Начинает реорганизацию с точки, где в последний раз завершила работу команда reorg. Используется, если в предыдущей команде reorg был задан лимит времени (time = количество_минут).

with time = количество_минут
Указывает, сколько минут должна выполняться команда reorg.

forwarded_rows
Реорганизует таблицу (индекс) так, чтобы в ней (нем) не было перемещенных строк.

compact
Совмещает функциональность команд reorg reclaim_space и reorg forwarded_rows , то есть освобождает пространство и устраняет перемещенные строки.

rebuild

Если указано имя таблицы, переписывает все строки таблицы на новые страницы, чтобы таблица размещалась в соответствии со своим кластерным индексом (если он существует), чтобы все ее страницы соответствовали текущим параметрам управления пространством, не было перемещенных строк и промежутков между строками на странице. Если указано имя индекса, команда reorg перестраивает этот индекс, оставляя таблицу доступной для чтения и обновления.

Примеры

Пример 1. Следующая команда делает неиспользуемое пространство в таблице titles доступным для будущего использования:

```
reorg reclaim_space titles
```

Пример 2. Команда делает неиспользуемое пространство страниц индекса titleind доступным для использования:

```
reorg reclaim_space titles titleind
```

Пример 3. Выполнение команды reorg compact для таблицы titles. Команда reorg обрабатывает таблицу с ее начала и продолжается в течение 120 минут. Если команда reorg завершается до истечения выделенного ей времени, она возвращается в начало таблицы и продолжает работу до окончания указанного интервала времени:

```
reorg compact titles with time = 120
```

Пример 4. Команда reorg compact запускается с точки, где остановилась предыдущая команда reorg compact, и выполняется в течение 30 минут:

```
reorg compact titles with resume, time = 30
```

Использование

- Таблица, указанная в команде reorg, должна использовать блокировку строк данных или страниц данных.
- Команду reorg нельзя выполнять внутри транзакции.
- Перед выполнением команды reorg rebuild нужно присвоить параметру базы данных select into/bulkcopy/pllsort значение true и выполнить команду checkpoint в этой базе данных.
- Для команды reorg rebuild требуется дополнительное дисковое пространство, равное размеру таблицы и ее индексов. Текущий объем пространства, занимаемого таблицей, можно выяснить с помощью процедуры sp_spaceused, а объем доступного пространства – с помощью процедуры sp_helpsegment.
- После выполнения команды reorg rebuild нужно сделать резервную копию базы данных, чтобы можно было сделать резервную копию журнала транзакций.

- Дополнительную информацию см. в книге *Руководство по системному администрированию*.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду reorg могут выполнять только системный администратор и владелец объекта.

См. также

Системные процедуры sp_chgattribute

return

Описание

Осуществляет безусловный выход из пакета или процедуры и дополнительно позволяет передать возвращаемое состояние. Операторы, идущие после команды return, не выполняются.

Синтаксис

return [*целочисленное_выражение*] [plan "abstract plan"]

Параметры

целочисленное_выражение

Целочисленное значение, возвращаемое процедурой. Хранимые процедуры могут возвращать целочисленное значение вызывающей процедуре или приложению.

plan "абстрактный план"

Задает абстрактный план, используемый для оптимизации запроса. Это может быть полный или частичный план, определенный на языке абстрактных планов. Планы могут указываться только для оптимизируемых операторов SQL, то есть для запросов, обращающихся к таблицам. Дополнительную информацию см. в главе 30, “Создание и использование абстрактных планов” книги *Руководство по настройке производительности*.

Примеры

Пример 1. Если в качестве параметра не задано ни одного имени пользователя, команда return завершит процедуру после вывода сообщения на экран пользователя. Если имя пользователя указано, то имена правил, созданных пользователем в текущей базе данных, извлекаются из соответствующих системных таблиц:

```
create procedure findrules @nm varchar(30) = null as
if @nm is null
begin
    print "You must give a user name"
    return
end
else
begin
    select sysobjects.name, sysobjects.id,
    sysobjects.uid
    from sysobjects, master..syslogins
        where master..syslogins.name = @nm
        and sysobjects.uid = master..syslogins.suid
        and sysobjects.type = "R"
end
```

Пример 2. Если из-за обновления средняя цена книг по бизнесу становится выше $15, команда return завершает выполнение пакета, не выполняя другие обновления таблицы titles:

```
print "Begin update batch"
update titles
    set price = price + $3
    where title_id = 'BU2075'
update titles
    set price = price + $3
    where title_id = 'BU1111'
if (select avg(price) from titles
      where title_id like 'BU%') > $15
begin
      print "Batch stopped; average price over $15"
    return
end
update titles
    set price = price + $2
      where title_id = 'BU1032'
```

Пример 3. Эта процедура создает два пользовательских кода состояний: значение 1 возвращается, если столбец contract содержит 1; значение 2 возвращается в остальных случаях (например, значение contract равно 0 или в таблице нет строк, в которых столбец title_id соответствует указанному условию):

```
create proc checkcontract @param varchar(11)
as
declare @status int
if (select contract from titles where title_id = @param)
= 1
    return 1
else
    return 2
```

Использование

- Возвращаемое состояние может применяться в последующих операторах пакета или процедуры, которые выполнили текущую процедуру, но должно быть представлено в следующей форме:

  ```
  execute @возвращаемое_значение = имя_процедуры
  ```

  Дополнительную информацию см. в описании команды execute.

- Adaptive Server резервирует 0 для отображения удачного возврата и отрицательные значения от -1 до -99 для обозначения различных причин сбоя. Если пользователем не задано возвращаемое значение, используется значение Adaptive Server. Возвращаемые состояния, определенные пользователем, не должны совпадать со значениями, зарезервированными сервером Adaptive Server (в частности, с номерами 0 и от -1 до -14, которые зарезервированы).

Таблица 7-30. Возвращаемые Adaptive Server значения ошибок

Значение	Смысл
0	Процедура выполнена без ошибок
-1	Объект отсутствует
-2	Ошибка типа данных
-3	Процесс остановлен в результате взаимоблокировки
-4	Ошибка полномочия
-5	Синтаксическая ошибка
-6	Различные ошибки пользователя
-7	Ошибка ресурса, например нехватка свободного пространства
-8	Нефатальная внутренняя проблема
-9	Было достигнуто системное предельное значение
-10	Фатальная внутренняя несогласованность
-11	Фатальная внутренняя несогласованность
-12	Таблица или индекс повреждены
-13	База данных повреждена
-14	Ошибка оборудования

Значения от -15 до -99 зарезервированы для будущего использования на Adaptive Server.

- Если во время выполнения возникло несколько ошибок, возвращается состояние с наибольшим абсолютным значением. Пользовательские возвращаемые значения всегда имеют более высокий приоритет, чем значения, возвращаемые Adaptive Server.
- Команда return может использоваться везде, где требуется выйти из пакета или процедуры. Возврат происходит немедленно и окончательно: операторы после команды return не выполняются.
- Хранимая процедура не может вернуть состояние NULL. Если процедура пытается вернуть значение NULL, например, командой return @состояние, где *@состояние* – NULL, выдается предупреждение и возвращается значение от 0 до -14.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду return по умолчанию могут выполнять все пользователи. Для этого не требуется никаких полномочий.

См. также

Команды begin...end, execute, if...else, while

revoke

Описание

Отзывает полномочия или роли у пользователей или ролей.

Синтаксис

Отзыв полномочий на доступ к объектам базы данных:

```
revoke [grant option for]
    {all [privileges] | список_полномочий}
    on { имя_таблицы [(список_столбцов)]
        | имя_представления[(список_столбцов)]
        | имя_хранимой_процедуры}
    from {public | список_имен | имя_роли}
    [cascade]
```

Для отзыва полномочия на создание объектов базы данных необходимо выполнить команду set proxy или set session authorization:

```
revoke {all [privileges] | список_команд }
    from {public | список_имен | имя_роли}
```

Отзыв роли у пользователя или другой роли:

```
revoke role {имя_роли [, имя_роли ...]} from
    {получатель_роли [, получатель_роли...]}
```

Параметры

all

Если команда используется для отзыва полномочий на доступ к объектам базы данных (первая форма синтаксиса), параметр all отзывает все полномочия на указанный объект. Все владельцы объектов могут отзывать полномочия на собственные объекты с помощью команды revoke all.

Отзывать полномочия на выполнение команд create (то есть использовать вторую форму синтаксиса) может только системный администратор и владелец базы данных. Если команда revoke all используется системным администратором, она отзывает все полномочия create (create database, create default, create procedure, create rule, create table и create view). Если команда revoke all используется владельцем базы данных, отзываются все полномочия create, кроме create database, и выдается информационное сообщение.

Ключевое слово all нельзя указывать при отзыве полномочий на выполнение команд set proxy и set session authorization.

список_полномочий

Список отзываемых полномочий. Если указано несколько полномочий, они должны быть разделены запятыми. В таблице приведены полномочия на доступ, которые могут быть предоставлены и отозваны для каждого типа объекта:

Объект	Что можно включить в список_полномочий
Таблица	select, insert, delete, update, references
Представление	select, insert, delete, update
Столбец	select, update, references Имена столбцов могут быть указаны в *списке_полномочий* или *списке_столбцов* (см. пример 2).
Хранимая процедура	execute

Полномочия могут быть отозваны только тем пользователем, который их предоставил.

список_команд

Перечень команд. Если указано несколько команд, они должны быть разделены запятыми. Список команд может содержать create database, create default, create procedure, create rule, create table, create view, set proxy или set session authorization. Отозвать полномочие на использование команды create database может только системный администратор и только из базы данных master.

Команды set proxy и set session authorization идентичны; единственное различие заключается в том, что set session authorization соответствует стандарту SQL, а set proxy является расширением, появившемся в Transact-SQL. Отзыв полномочий на выполнение команды set proxy или set session authorization означает запрет выдавать себя за другого пользователя сервера. Полномочия на выполнение команд set proxy или set session authorization может отозвать только администратор по безопасности системы и только из базы данных master.

имя_таблицы

Имя таблицы, полномочия на использование которой отзываются. Эта таблица должна быть в текущей базе данных. В операторе revoke можно указать только один объект.

список_столбцов

Список столбцов (через запятую), к которым применяются полномочия. Если столбцы указаны, можно отзывать только полномочия на выполнение команд select и update.

имя_представления

Имя представления, полномочия на использование которого отзываются. Представление должно быть в текущей базе данных. Для каждого оператора revoke можно указать только один объект.

имя_хранимой_процедуры
Имя хранимой процедуры, полномочия на использование которой отзываются. Эта хранимая процедура должна быть в текущей базе данных. В операторе revoke может быть указан только один объект.

public
Все пользователи. При предоставлении полномочий на доступ к объекту группа public не включает владельца этого объекта. Применительно к полномочиям на создание объекта или полномочиям set proxy группа public не включает владельца базы данных. Группе public, а также другим группам и ролям нельзя предоставлять полномочия командой grant с параметром with grant option.

список_имен
Список имен пользователей и/или имен групп, разделенных запятыми.

role
Имя системной или пользовательской роли. Команда revoke role используется для отзыва предоставленных ролей у других ролей или пользователей.

имя_роли
Имя системной или пользовательской роли. Позволяет отзывать полномочия у всех пользователей, которым были предоставлены определенные роли. Имя роли может быть системной или пользовательской ролью, созданной администратором безопасности оператором create role. Роль любого типа может быть предоставлена пользователю командой grant role. Кроме того, для предоставления системных ролей можно использовать процедуру sp_role.

получатель_роли
Имя системной или пользовательской роли, или имя пользователя, у которого отзывается роль.

grant option for
Отзывает полномочия, предоставленные с параметром with grant option, чтобы пользователи, указанные в *списке_имен*, больше не могли передавать указанные полномочия другим пользователям. Если такие пользователи уже предоставили полномочия другим пользователям, необходимо применить параметр cascade, чтобы отозвать полномочия у этих пользователей. Пользователь, указанный в *списке_имен*, сохраняет полномочие на доступ к объекту, но больше не может предоставлять полномочия доступа другим пользователям. Инструкция grant option for применяется только к полномочиям на доступ к объектам, но не к полномочиям на создание объектов.

cascade

Отзывает указанные полномочия на доступ к объекту для всех пользователей, получивших их ранее от пользователя, полномочия которого отозваны. Применяется только к полномочиям на доступ к объектам, но не к полномочиям на создание объектов. Если выполнить команду revoke без grant option for, отзываются также полномочия других пользователей, полученные от пользователя, чьи полномочия отозваны, то есть произойдет каскадный отзыв.

Примеры

Пример 1. Отзыв полномочий insert и delete для таблицы titles у пользователя Mary и группы sales:

```
revoke insert, delete
on titles
from mary, sales
```

Пример 2. Два способа отозвать полномочия update для столбцов price и advance в таблице titles у всех пользователей:

```
revoke update
on titles (price, advance)
from public
```

или:

```
revoke update (price, advance)
on titles
from public
```

Пример 3. Команда отзывает у пользователей Mary и John полномочие на выполнение команд create database и create table. Эту команду может выполнить только системный администратор и только из базы данных master, поскольку отзывается полномочие на выполнение create database. Полномочие на использование команды create table отзывается у пользователей Mary и John только в базе данных master:

```
revoke create database, create table from mary, john
```

Пример 4. Отзыв у пользователей Harry и Billy полномочий на выполнение команд set proxy и set session authorization, позволяющих работать под именем другого пользователя на сервере:

```
revoke set proxy from harry, billy
```

Пример 5. Отзыв у пользователей с ролью sso_role полномочий на выполнение команд set proxy и set session authorization:

```
revoke set session authorization from sso_role
```

Пример 6. Отзыв у пользователей с ролью vip_role права работать под именем другого пользователя на сервере. Роль vip_role должна быть определена администратором безопасности системы командой create role:

```
revoke set proxy from vip_role
```

Пример 7. Отзыв у пользователя Mary всех полномочий на создание объектов в текущей базе данных:

```
revoke all from mary
```

Пример 8. Отзыв у пользователя Mary всех полномочий на доступ к объектам для таблицы titles:

```
revoke all on titles from mary
```

Пример 9. Два способа отозвать у пользователя Tom полномочия на создание ограничений целостности для другой таблицы, ссылающихся на столбцы price и advance в таблице titles:

```
revoke references
on titles (price, advance)
from tom
```

или:

```
revoke references (price, advance)
on titles
from tom
```

Пример 10. Отзыв полномочий на выполнение процедуры new_sproc у всех пользователей с ролью operator:

```
revoke execute on new_sproc from oper_role
```

Пример 11. Отзыв у пользователя John прав на предоставление другим пользователям полномочий на выполнение команд insert, update и delete с таблицей authors. Эти полномочия также отзываются у других пользователей, которым они были предоставлены пользователем John:

```
revoke grant option for
insert, update, delete
on authors
from john
cascade
```

Пример 12. Отзыв полномочий роли doctor_role у пользователей с ролью specialist_role:

```
revoke role doctor_role from specialist_role
```

Пример 13. Отзыв полномочий ролей doctor_role и surgeon_role у пользователей с ролью specialist_role и intern_role, а также у пользователей Mary и Tom:

```
revoke role doctor_role, surgeon_role from
specialist_role, intern_role, mary, tom
```

Использование

- Дополнительную информацию о полномочиях см. в описании команды grant.
- Отзывать полномочия можно только для объектов в текущей базе данных.
- Пользователь может отозвать только те полномочия, которые он сам предоставил.
- Нельзя отзывать роль у пользователя, пока он работает в системе.
- Команды grant и revoke зависят от порядка выполнения. Если возникает конфликт, выполняется команда, которая была запущена последней.
- Слово to можно использовать вместо слова from в синтаксисе revoke.
- Если в операторе revoke не указан параметр grant option for, то у пользователя отзываются и полномочия with grant option, и указанные полномочия на доступ к объектам. Кроме того, если пользователь предоставил указанные полномочия другим пользователям, все эти полномочия также отзываются. Другими словами, команда revoke имеет каскадный эффект.
- Оператор grant добавляет одну строку в системную таблицу sysprotects для каждого пользователя, группы или роли, получающих полномочия. Если затем с помощью команды revoke отозвать полномочия у пользователя или группы, эта строка будет удалена из таблицы sysprotects. Если полномочия отзываются только у некоторых членов группы, а не у всей группы, которой они были предоставлены, первоначальная строка сохранится и будет добавлена новая строка, соответствующая отзыву.
- По умолчанию команду create trigger могут выполнять все пользователи. При отзыве у пользователя полномочий на создание триггеров в таблицу sysprotects добавляется строка, соответствующая отзыву у этого пользователя. Чтобы разрешить этому пользователю выполнять команду create trigger, нужно выполнить две команды grant. Первая команда удаляет строку отзыва из sysprotects; вторая добавляет строку о предоставлении полномочия. Если у пользователя было отозвано полномочие на создание триггеров, то он не сможет созда-

вать триггеры даже для собственных таблиц. Отзыв у пользователя полномочий на создание триггеров действует только на базу данных, где выполняется команда revoke.

Использование параметра *cascade*

- Команда revoke grant option лишает пользователя права предоставлять указанные полномочия другим пользователям, но не отзывает само полномочие у этого пользователя. Если пользователь предоставил это полномочие другим, необходимо использовать параметр cascade, иначе будет выдано сообщение об ошибке, и команда revoke не будет выполнена.

  Например, пусть у пользователя Bob отзывается полномочие для таблицы titles, которое было ему предоставлено с указанием параметра with grant option:

  ```
  revoke grant option for select
  on titles
  from bob
  cascade
  ```

  - Если Bob не предоставлял это полномочие другим пользователям, команда отменяет его право делать это, но сам Bob сохранит полномочие на использование команды select с таблицей titles.
  - Если Bob предоставил это полномочие другим пользователям, то необходимо использовать параметр cascade. Иначе будет выдано сообщение об ошибке, и команда revoke не будет выполнена. Благодаря параметру cascade будут отозваны полномочия на использование команды select у всех пользователей, которым они были предоставлены пользователем Bob, вместе с возможностью предоставлять это полномочие другим пользователям.

- Нельзя использовать revoke с параметром cascade для отзыва полномочий, предоставленных владельцем таблицы. Например, пусть владелец таблицы (UserA) предоставил полномочия другому пользователю (UserB) следующей командой:

  ```
  create table T1 (...)
  grant select on T1 to UserB
  ```

  В этом случае системный администратор не может отозвать привилегии у пользователя UserB следующей командой revoke с параметром cascade:

  ```
  revoke select on T1 from UserA cascade
  ```

  Этот оператор отзывает привилегии select у владельца таблицы, но не отзывает их у пользователя UserB.

По умолчанию все операции языка управления данными (data manipulation language, DML) неявно отзываются у всех пользователей, кроме владельца таблицы. Поскольку таблица sysprotects не содержит записей о том, что владелец таблицы предоставил, а затем отозвал привилегии, параметр cascade не даст эффекта.

Чтобы отозвать полномочие select у пользователя UserB, это нужно сделать явно.

Отзыв полномочий на команды *set proxy* и *set session authorization*

- Отзывать полномочия set proxy и set session authorization и роли может только администратор по безопасности и только из базы данных master.
- Команды set proxy и set session authorization идентичны, за одним исключением: set session authorization соответствует стандарту SQL. Если по тем или иным причинам необходимо применять только стандартные команды и синтаксис SQL, используется команда set session authorization.
- Команда revoke all *не* затрагивает полномочия set proxy или set session authorization.

Отзыв полномочий у ролей, пользователей и групп

- Полномочия, предоставленные ролям, приоритетнее полномочий, данных отдельным пользователям или группам. Поэтому если отозвать полномочие у пользователя, которому была дана роль, обладающая тем же полномочием, пользователь сохранит свое полномочие. Например, пусть пользователю John предоставлена роль System Security Officer (администратор по безопасности), а роли sso_role было дано полномочие для таблицы sales. Если индивидуальное полномочие пользователя John для таблицы sales отозвано, он по-прежнему может получать доступ к этой таблице, поскольку полномочия его роли приоритетнее индивидуальных полномочий.
- Если полномочие отзывается у всех пользователей (группы public) или у другой группы, оно отзывается и у пользователей, которым оно было предоставлено индивидуально.
- Благодаря тому, что пользователей базы данных можно объединять в группы, одним оператором можно предоставлять или отзывать полномочия для нескольких пользователей. Пользователь всегда является участником группы public (группа по умолчанию) и может быть участником лишь еще одной группы. Сценарий установки Adaptive Server назначает набор полномочий группе public.

Для создания групп используется процедура sp_addgroup, а для удаления групп – процедура sp_dropgroup. Для добавления новых пользователей в группу используется процедура sp_adduser, а для изменения членства пользователя в группе – процедура sp_changegroup. Для показа членов группы используется процедура sp_helpgroup.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Доступ к объектам базы данных. Команду revoke для объектов базы данных по умолчанию могут выполнять только владельцы объектов. Владелец объекта может отозвать у других пользователей полномочие на свои объекты базы данных.

Выполнение команд. Полномочие create database может быть отозвано только системным администратором, причем он может сделать это только из базы данных master. Полномочие create trigger может быть отозвано только администратором по безопасности.

Прокси-авторизация и авторизация сеансов. Полномочия set proxy и set session authorization могут быть отозваны только администратором по безопасности, причем он может сделать это только из базы данных master.

Роли. Отзыв ролей можно выполнять только из базы данных master. Только администратор по безопасности вправе отзывать роли sso_role и oper_role, а также пользовательские роли у других пользователей или ролей. Только системные администраторы вправе отзывать роль sa_role у пользователя или роли. Только пользователь, обладающий ролями sa_role и sso_role вправе отзывать роль, включающую роль sa_role.

См. также

Команды grant, setuser, set

Функции proc_role

Системные процедуры sp_activeroles, sp_adduser, sp_changedbowner, sp_changegroup, sp_displaylogin, sp_displayroles, sp_dropgroup, sp_dropuser, sp_helpgroup, sp_helprotect, sp_helpuser, sp_modifylogin, sp_role

rollback

Описание

Выполняет откат пользовательской транзакции до именованной точки сохранения внутри этой транзакции или до начала транзакции.

Синтаксис

rollback [tran | transaction | work]
[*имя_транзакции* | *имя_точки_сохранения*]

Параметры

tran | transaction | work

Означает, что нужно произвести откат транзакции, то есть отменить проделанную в ходе этой транзакции работу. Если задано одно из ключевых слов tran, transaction или work, можно указать также параметр *имя_транзакции* или *имя_точки_сохранения*.

имя_транзакции

Имя самой внешней транзакции. Оно должно соответствовать правилам именования идентификаторов.

имя_точки_сохранения

Имя, присвоенное точке сохранения в операторе save transaction. Оно должно соответствовать правилам именования идентификаторов.

Примеры

Откат транзакции:

```
begin transaction
delete from publishers where pub_id = "9906"
rollback transaction
```

Использование

- Оператор rollback transaction, в котором не указано ни *имя_транзакции*, ни *имя_точки_сохранения*, производит откат пользовательской транзакции к началу самой внешней транзакции.
- Оператор rollback transaction имя_транзакции выполняет откат пользовательской транзакции к началу транзакции с указанным именем. Хотя можно организовывать вложенные транзакции, откат можно делать только для самой внешней транзакции.
- Оператор rollback transaction имя_точки_сохранения производит откат пользовательской транзакции к соответствующей точке сохранения, созданной оператором save transaction имя_точки_сохранения.

Ограничения

- Если в данный момент нет активной транзакции, операторы commit и rollback не выполняют никаких действий.
- Команда rollback должна выполняться внутри транзакции. Нельзя делать откат транзакции после выполнения оператора commit.

Откат транзакции целиком

- Команда rollback без указания имени точки сохранения отменяет транзакцию целиком. Отменяются результаты действия всех операторов и процедур, входящих в транзакцию.
- Если в операторе rollback не указаны ни *имя_точки_сохранения*, ни *имя_транзакции*, откат транзакции производится к первому оператору begin transaction в пакете. Это относится также и к транзакциям, которые были начаты неявным оператором begin transaction в режиме связанных транзакций.

Откат к точке сохранения

- Для отмены части транзакции в операторе rollback должно быть указано *имя_точки_сохранения*. Точка сохранения представляет собой метку, установленную пользователем внутри транзакции с помощью команды save transaction. Все результаты действия операторов и процедур между точкой сохранения и оператором rollback отменяются.

  После отката к точке сохранения транзакция может быть продолжена (путем выполнения любых операторов SQL после данного оператора rollback) до завершения оператором commit или отменена полностью с помощью оператора rollback без указания точки сохранения. На число точек сохранения внутри транзакции не накладывается никаких ограничений.

Откаты транзакций в триггерах и хранимых процедурах

- В триггерах или хранимых процедурах команды rollback без указания имени транзакции или точки сохранения делают откат всех операторов до первого явного или неявного оператора begin transaction в пакете, из которого была вызвана процедура или запущен триггер.
- Если триггер содержит команду rollback без указания точки сохранения, откат приводит к аварийному завершению всего пакета. Операторы пакета, следующие за оператором, вызвавшим откат, не выполняются.
- Вызов удаленной процедуры выполняется независимо от транзакции, в состав которой он входит. В стандартной транзакции (то есть не использующей двухфазную фиксацию с применением библиотеки Open Client™ DB-Library) команды, выполняемые удаленным сервером через вызов удаленной процедуры, не могут быть отменены оператором rollback и не зависят от выполнения оператора commit .

- Полная информация об использовании операторов управления транзакциями и о влиянии оператора rollback на хранимые процедуры, триггеры и пакеты содержится в книге *Transact-SQL User's Guide*.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Расширения Transact-SQL Формы записи оператора rollback transaction и rollback tran, а также использование имени транзакции.

Полномочия

Команду rollback по умолчанию могут выполнять все пользователи. Для этого не требуется никаких полномочий.

См. также

Команды begin transaction, commit, create trigger, save transaction

rollback trigger

Описание

Отменяет результаты действий, выполненных в триггере (включая операцию изменения данных, запустившую триггер), и может (но не обязательно) выполнять оператор raiserror.

Синтаксис

rollback trigger
[with *команда_raiserror*]

Параметры

with команда_raiserror

Задает команду raiserror, которая выводит определенное пользователем сообщение об ошибке и регистрирует возникновение ошибки путем установки системного флага. Это позволяет при выполнении оператора rollback trigger уведомлять клиента об ошибке, чтобы состояние транзакции при ошибке отражало откат триггера. Информацию о синтаксисе и правилах написания команды_raiserror см. в описании этой команды.

Примеры

Откат триггера и выдача определенного пользователем сообщения об ошибке 25002:

```
rollback trigger with raiserror 25002
    "title_id does not exist in titles table."
```

Использование

- При выполнении оператора rollback trigger сервер Adaptive Server прекращает выполнение текущей команды и выходит из триггера.
- Если триггер, в котором выполняется оператор rollback trigger, вложен в другие триггеры, сервер Adaptive Server отменяет все изменения, сделанные в этих триггерах, включая операцию изменения данных, которая запустила первый триггер.
- Сервер Adaptive Server игнорирует оператор rollback trigger, если он выполняется вне триггера, и не выполняет связанного с ним оператора raiserror. Однако оператор rollback trigger, выполняемый вне триггера, но внутри транзакции, создает ошибку, которая вынуждает сервер Adaptive Server выполнить откат транзакции и аварийно завершить текущий пакет операторов.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду rollback trigger по умолчанию могут выполнять все пользователи. Для этого не требуется никаких полномочий.

См. также

Операторы create trigger, raiserror, rollback

save transaction

Описание — Устанавливает точку сохранения внутри транзакции.

Синтаксис — save transaction *имя_точки_сохранения*

Параметры — *имя_точки_сохранения*

Имя, присваиваемое точке сохранения. Оно должно соответствовать правилам именования идентификаторов.

Примеры — После изменения значений столбца royaltyper для двух авторов вставляется точка сохранения percentchanged, затем подсчитывается, как повышение цены книги на 10 процентов повлияет на авторский гонорар. Для транзакции производится откат к точке сохранения посредством оператора rollback transaction:

```
begin transaction royalty_change

update titleauthor
set royaltyper = 65
from titleauthor, titles
where royaltyper = 75
and titleauthor.title_id = titles.title_id
and title = "The Gourmet Microwave"

update titleauthor
set royaltyper = 35
from titleauthor, titles
where royaltyper = 25
and titleauthor.title_id = titles.title_id
and title = "The Gourmet Microwave"

save transaction percentchanged

update titles
set price = price * 1.1
where title = "The Gourmet Microwave"

select (price * total_sales) * royaltyper
from titles, titleauthor
where title = "The Gourmet Microwave"
and titles.title_id = titleauthor.title_id

rollback transaction percentchanged

commit transaction
```

Использование

- Полная информация об использовании операторов управления транзакциями содержится в книге *Transact-SQL User's Guide*.
- Точка сохранения представляет собой определенную пользователем метку внутри транзакции, которая позволяет отменять части транзакции. Оператор rollback имя_точки_сохранения делает откат изменений к указанной точке сохранения. Отменяются результаты действия всех операторов и процедур между точкой сохранения и оператором rollback.

  Предшествующие точке сохранения операторы не отменяются, но и не фиксируются. После отката к точке сохранения транзакция продолжает выполнять операторы. Оператор rollback без указания точки сохранения выполняет откат всей транзакции. Оператор commit завершает транзакцию.
- Если транзакции являются вложенными, оператор save transaction создает точку сохранения только для самой внешней транзакции.
- На число точек сохранения внутри транзакции не накладывается никаких ограничений.
- Если в операторе rollback не указаны ни *имя_точки_сохранения*, ни *имя_транзакции*, делается откат всех команд до первого оператора begin transaction в пакете и транзакция отменяется целиком.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду save transaction по умолчанию могут выполнять все пользователи. Для этого не требуется никаких полномочий.

См. также

Команды begin transaction, commit, rollback

select

Описание

Извлекает строки из объектов базы данных.

Синтаксис

select ::=
    select [all | distinct] *список_выборки*
    [*инструкция_into*]
    [*инструкция_from*]
    [*инструкция_where*]
    [*инструкция_group_by*]
    [*инструкция_having*]
    [*инструкция_order_by*]
    [*инструкция_compute*]
    [*инструкция_read_only*]
    [*инструкция_isolation*]
    [*инструкция_browse*]
    [*инструкция_plan*]

список_выборки ::=

Примечание. Подробную информацию о *списке_выборки* см. в описании параметров.

инструкция_into ::=
    into [[*база_данных*.]*владелец*.]*имя_таблицы*
    [lock {datarows | datapages | allpages }]
    [with *параметр_into*[, *параметр_into*] ...]

    параметр_into ::=
        | max_rows_per_page = *количество_строк*
        | exp_row_size = *количество_байтов*
        | reservepagegap = *количество_страниц*
        | identity_gap = *шаг*
        [existing table *имя_таблицы*]
        [[external *тип*] at "*путь*"
        [column delimiter *разделитель*]]

инструкция_from ::=
    from *ссылка_на_таблицу* [, *ссылка_на_таблицу*]...

    ссылка_на_таблицу::=
        имя_представления_или_таблицы | *соединение_ANSI*

        имя_представления_или_таблицы ::=
            [[*база_данных*.]*владелец*.] {{*имя_таблицы* |
        имя_представления}
            [as] [*псевдоним*]
            [index {*имя_индекса* | *имя_таблицы* }]
            [parallel [*степень_параллелизма*]]
            [prefetch size][lru | mru]}
        [holdlock | noholdlock]

```
                [readpast]
                [shared]

                соединение_ANSI ::=
                    ссылка_на_таблицу тип_соединения join
                    ссылка_на_таблицу
                            условия_соединения
                        тип_соединения::= inner | left [outer] | right [outer]
                        условия_соединения ::= on условия_поиска

инструкция_where ::=
        where условия_поиска

инструкция_group_by ::=
        group by [all] выражение_без_агрегатов
            [, выражение_без_агрегатов]...

инструкция_having ::=
        where условия_поиска

инструкция_order_by ::=
        order by инструкция_sort [, инструкция_sort]...

        инструкция_sort ::=
            {
[[[база_данных.]владелец.]{имя_таблицы.|имя_представления.}]имя_ст
олбца
            | номер_в_списке_выборки
            | выражение }
            [asc | desc]

инструкция_compute ::=
        compute строка_агрегат(имя_столбца)
            [, строка_агрегат(имя_столбца)]...
        [by имя_столбца [, имя_столбца]...]

инструкция_read_only ::=
        for {read only | update [of список_имен_столбцов]}

инструкция_isolation ::=
        at isolation
            { read uncommitted | 0 }
            | { read committed | 1 }
            | { repeatable read | 2 }
            | { serializable | 3 }

инструкция_browse ::=
        for browse

инструкция_plan ::=
        plan "абстрактный план"
```

Параметры

all

Включает все строки в набор результатов. all – настройка по умолчанию.

distinct

Включает в набор результатов только уникальные строки. Ключевое слово distinct должно быть первым словом в списке выборки; в режиме просмотра оно игнорируется.

Для ключевого слова distinct значения NULL считаются равными друг другу. Поэтому будет выбрана только одна строка со значениями NULL, независимо от того, сколько таких строк удовлетворяет остальным условиям запроса.

список_выборки

Включает в себя один или несколько следующих элементов:

- символ “*”, означающий, что будут выбраны все столбцы (в порядке, определенном в команде create table).
- Список имен столбцов (в произвольном порядке). Если нужно выбрать существующий столбец IDENTITY, можно вместо фактического имени столбца указать ключевое слово syb_identity, при необходимости уточненное именем таблицы.
- Добавление столбца IDENTITY к таблице результатов:

  ```
  имя_столбца = identity(точность)
  ```

- Псевдоним, использующийся вместо заголовка столбца по умолчанию (которым является имя столбца). Имеет одну из следующих форм:

  ```
  заголовок_столбца = имя_столбца
  имя_столбца заголовок_столбца
  имя_столбца as заголовок_столбца
  ```

  В любой из этих форм заголовок столбца может быть заключен в кавычки. Заголовок столбца нужно заключить в кавычки, если он не является допустимым идентификатором (то есть либо является зарезервированным словом, либо начинается со специального символа, либо содержит пробелы или знаки пунктуации).
- Выражение (имя столбца, константа, функция или любая комбинация имен столбцов, констант и функций, соединенных арифметическими или поразрядными операциями, или подзапрос).
- Встроенная функция или агрегат.
- Любая комбинация вышеперечисленных элементов.

В *списке_выборки* также можно присваивать значения переменным с помощью следующего синтаксиса:

```
@переменная = выражение
    [, @переменная = выражение ...]
```

Нельзя совмещать присваивание значений переменным с каким-либо другим вариантом синтаксиса *списка_выборки*.

into
: Создает новую таблицу на основе столбцов, указанных в списке выборки, и строк, выбранных инструкцией where. См. подраздел "Использование команды select into" в этом разделе.

lock datarows | datapages | allpages
: Указывает схему блокировки, которая будет использоваться для таблицы, создаваемой командой select into. По умолчанию применяется настройка параметра конфигурации lock scheme, заданная на уровне сервера.

max_rows_per_page
: Ограничивает количество строк на страницах данных таблицы, создаваемой с помощью команды select into. В отличие от параметра fillfactor, значение max_rows_per_page сохраняется при вставке или удалении данных. Параметр max_rows_per_page не поддерживается в таблицах с блокировкой только данных.

existing table имя_таблицы
: Указывает, что данные выбираются в прокси-таблицу. Этот синтаксис select into нельзя использовать ни с какими другими типами таблиц. Столбцы, указанные в списке выборки, должны соответствовать столбцам прокси-таблицы по типу, длине и количеству.

at "путь"
: Указывает полный путь к внешнему файлу, в который записываются выбранные данные. Параметр at можно указывать только при выборе в прокси-таблицу.

external [table | file]
: file указывает, что внешний объект является файлом, а table – что таблицей. Если в команде select into не указано ни file, ни table, то считается, что используется таблица.

column delimeter "*разделитель*"
: Указывает разделитель, используемый для отделения столбцов друг от друга. Если разделитель не указан, то в команде select into разделителем является символ табуляции.

exp_row_size = количество_байтов
: Указывает ожидаемый размер строки для таблицы, создаваемой с помощью команды select into. Применяется только для схем блокирования datarows и datapages и только для таблиц со строками переменной длины. Допустимые значения – 0, 1 и любое значение между минимальной и максимальной длиной строки таблицы. Значение 0 (являющееся значением по умолчанию) означает, что используется значение по умолчанию, заданное на уровне сервера.

reservepagegap = количество_страниц
: Указывает отношение числа заполненных страниц к числу страниц, которые должны остаться пустыми при выделении экстентов для хранения данных командой select into. Этот параметр можно использовать только в команде select into. На каждые *количество_страниц* страниц оставляется одна пустая страница для расширения таблицы в будущем. Допустимые значения – от 0 до 255. Значение по умолчанию – 0.

readpast
: Указывает, что запрос должен пропускать строки с монопольными блокировками, не ожидая их освобождения и не выдавая сообщение.

with identity_gap
: Указывает интервал между значениями столбца IDENTITY в таблице. Для данной таблицы это значение имеет приоритет над системной настройкой.

шаг
: Величина интервала между значениями IDENTITY.

  Таблица, создаваемая командой select into, не наследует настройку identity gap от родительской таблицы. Вместо этого в новой таблице применяется параметр identity burning set factor. В команде select into можно задать для новой таблицы другое значение интервала identity_gap. Интервал между значениями IDENTITY для новой таблицы может быть таким же, как и в родительской таблице, а может и отличаться от него.

from
: Указывает, какие таблицы и представления используются в команде select. Это ключевое слово обязательно, за исключением случая, когда список выборки не содержит имен столбцов, а только константы и арифметические выражения, например:

```
select 5 x, 2 y, "the product is", 5*2 Result

x           y                         Result
----------- ----------- -------------- -----------
          5           2 the product is          10
```

В запросе может быть указано не более 50 таблиц и 14 рабочих таблиц (например, создаваемых агрегатными функциями). В этот 50-табличный лимит входят:

- таблицы (или представления на таблицах), указанные в инструкции from;
- все ссылки на одну и ту же таблицу (одна таблица может быть указана более одного раза, например, в рефлексивных соединениях);
- таблицы, указанные в подзапросах;
- таблицы, создаваемые с помощью инструкции into;
- базовые таблицы, на которые ссылаются представления, содержащиеся в инструкции from.

имя_представления, *имя_таблицы*

Список таблиц и представлений, используемых в команде select. Если таблица или представление находится в другой базе данных, нужно указать имя базы данных, а если в этой базе данных существует несколько таблиц или представлений с указанным именем, нужно указать владельца. По умолчанию *владельцем* является текущий пользователь, а *базой_данных* – текущая база данных.

Если список содержит несколько таблиц или представлений, их имена должны быть разделены запятыми. Порядок, в котором указаны таблицы или представления после ключевого слова from, не влияет на результат.

В одной команде можно делать запросы к таблицам, находящимся в разных базах данных.

Таблицам и представлениям можно назначать псевдонимы (сопоставляющие имена). Псевдонимы делают текст запроса более понятным, а также позволяют указывать одну и ту же таблицу несколько раз (это нужно, в частности, в запросах, содержащих рефлексивные соединения или подзапросы). Чтобы назначить псевдоним, укажите имя таблицы или представления, затем пробел и введите псевдоним, например:

```
select pub_name, title_id
    from publishers pu, titles t
    where t.pub_id = pu.pub_id
```

Все другие ссылки на эту таблицу или представление (например, в инструкции where) должны использовать этот псевдоним. Псевдонимы не могут начинаться с цифры.

index имя_индекса

Указывает индекс, который нужно использовать для доступа к таблице *имя_таблицы*. Этот параметр нельзя использовать при запросе к представлению, но его можно указать в инструкции select команды create view.

parallel

Указывает, что будет применено параллельное сканирование секций или сканирование индекса, если конфигурация Adaptive Server допускает параллельную обработку.

степень_параллелизма

Задает число рабочих процессов, которые будут параллельно сканировать таблицу или индекс. Если этот параметр равен 1, запрос выполняется последовательно.

prefetch размер

Указывает объем ввода-вывода (в килобайтах) для таблиц, кэши которых сконфигурированы для больших объемов ввода-вывода. Этот параметр нельзя использовать при запросе к представлению, но можно указать в инструкции select команды create view. Процедура sp_helpcache показывает допустимые объемы для кэша, с которым связан объект, или для кэша по умолчанию. Для настройки размера кэша данных используйте процедуру sp_cacheconfigure.

Возможные значения *размера* для предварительной выборки, указываемого после ключевого слова prefetch, – 2 КБ или любое кратное двум число на логической странице размером до 16 КБ. Параметры размера prefetch в килобайтах приведены ниже:

Размер логической страницы	Размер для предварительной выборки
2	2, 4, 8 16
4	4, 8, 16, 32
8	8, 16, 32, 64
16	16, 32, 64, 128

Размер, указанный в запросе после ключевого слова prefetch – это просто подсказка. Чтобы указанный размер можно было использовать, нужно сконфигурировать кэш данных этого размера. Если кэш данных заданного размера не сконфигурирован, будет использоваться размер prefetch, заданный по умолчанию.

Использование параметра prefetch для удаленных серверов невозможно, если включены службы Component Integration Services.

lru | mru

Указывает стратегию замены значений в буфере, которая будет использоваться для таблицы. Если указать lru , то оптимизатор будет считывать таблицу в кэш по алгоритму MRU/LRU (most recently used/least recently used, прочитанный позже/раньше всех остальных). Если указать mru, буфер будет сразу удаляться из кэша и заменяться на следующий буфер для таблицы. Этот параметр нельзя использовать при запросе к представлению, но его можно указать в инструкции select команды create view.

holdlock

Делает разделяемую блокировку указанной таблицы или представления более ограничительной, удерживая ее до завершения транзакции (вместо того, чтобы снимать разделяемую блокировку как только страница данных больше не требуется, независимо от того, завершена ли транзакция).

Параметр holdlock применяется только к таблице или представлению, для которого он указан, и действует только на протяжении транзакции, содержащей данную команду select. Если была выполнена команда set transaction isolation level 3, то установка holdlock неявно применяется ко всем командам select внутри транзакции. Ключевое слово holdlock недопустимо в команде select, содержащей параметр for browse. В одном запросе нельзя одновременно указывать параметры holdlock и noholdlock.

Если службы Component Integration Services включены, параметр holdlock нельзя указывать для удаленных серверов.

noholdlock

Не дает серверу удерживать блокировки, полученные во время выполнения этой команды select (независимо от действующего уровня изоляции транзакций). В одном запросе нельзя одновременно указывать параметры holdlock и noholdlock.

shared

Указывает Adaptive Server использовать разделяемую блокировку (вместо блокировки обновления) для указанной таблицы или представления. Это позволяет другим клиентам получить блокировку обновления на эту таблицу или представление. Ключевое слово shared можно использовать только с инструкцией select, входящей в состав команды declare cursor, например:

```
declare shared_crsr cursor
for select title, title_id
from titles shared
where title_id like "BU%"
```

Ключевое слово holdlock можно использовать вместе с ключевым словом shared после каждого имени таблицы или представления, (при этом holdlock должно быть указано перед shared).

соединение ANSI

Внутреннее или внешнее соединение, использующее синтаксис ANSI. Таблицы, участвующие в соединении, должны быть указаны в инструкции from.

inner

Указывает, что будет выполнено внутреннее соединение (включающее только строки внутренней и внешней таблиц, удовлетворяющие условиям инструкции on). Результирующий набор запроса с внутренним соединением не содержит строк внешней таблицы, не удовлетворяющих условиям инструкции on.

outer

Включает все строки внешней таблицы, независимо от того, удовлетворяют ли они условиям инструкции on. Если строки не удовлетворяют условиям инструкции on, значения из внутренней таблицы сохраняются в итоговой таблице как значения NULL. То, какие строки будут включены в результирующий набор внешнего соединения ANSI, определяется инструкцией where.

left

При левом соединении выбираются все строки таблицы, указанной слева от инструкции join. Левую таблицу называют внешней таблицей, или таблицей с сохранением строк.

В приведенных ниже запросах T1 – внешняя таблица, а T2 – внутренняя таблица:

```
T1 left join T2
T2 right join T1
```

right

При правом соединении выбираются все строки таблицы, указанной справа от инструкции join (см. пример выше)

условия_поиска

используется для настройки условий для поиска строк. Условие поиска может включать имена столбцов, выражения, арифметические операторы, операторы сравнения, ключевые слова not, like, is null, and, or, between, in, exists, any и all, подзапросы, выражения case и любые комбинации этих элементов. Подробности см. в разделе “where” на стр. 767.

group by

Вычисляет значение для каждой группы. Эти значения отображаются в наборе результатов не в виде новых строк, а в виде новых столбцов.

В стандартном языке SQL в команде, содержащей инструкцию group by, все элементы списка выборки должны либо иметь фиксированное значение в каждой строке группы, либо являться агрегатными функциями, которые вычисляют скалярное значение для каждой группы. В Transact-SQL отсутствуют подобные ограничения на элементы в списке выборки. Кроме того, Transact-SQL позволяет выполнять группировку по любому выражению (кроме псевдонимов столбцов), тогда как в стандартном языке SQL разрешена группировка только по столбцам.

В таблице 7-31 перечислены агрегатные функции, которые можно использовать с инструкцией group by (*выражение* – это почти всегда имя столбца):

Таблица 7-31. Результаты использования агрегатных функций в инструкции group by

Агрегатная функция	Результат
sum([all | distinct] выражение)	Сумма значений числового столбца.
avg([all | distinct] выражение)	Среднее значений числового столбца.
count([all | distinct] выражение)	Количество отличных от NULL (различающихся) значений в столбце.
COUNT(*)	Количество выбранных строк.
max(выражение)	Наибольшее значение в столбце.
min(выражение)	Наименьшее значение в столбце.

Подробности см. в разделе “group by и having” на стр. 576.

Таблицу можно сгруппировать по любой комбинации столбцов, то есть группы могут быть вложены друг в друга. В инструкции group by можно указывать имя столбца, выражение или номер, представляющий позицию элемента в списке выборки, но не заголовок столбца.

group by all

Включает в набор результатов все группы, даже те, в которых нет строк, соответствующих условиям поиска. Пример см. в разделе “group by и having” на стр. 576.

выражение_без_агрегатов

Выражение, в котором нет агрегатов.

having
Задает условия для инструкции group by аналогично тому, как инструкция where задает условия для инструкции select. Количество условий, которые можно включить в эту инструкцию, не ограничено.

Инструкцию можно использовать having без инструкции group by.

Если к каким-либо столбцам списка выборки не применяются агрегатные функции и эти столбцы не включены в инструкцию group by (что запрещено в стандартном SQL), то инструкции having и where будут иметь немного разную семантику.

В этой ситуации инструкция where накладывает фильтр на строки, включаемые в агрегатное вычисление, но не накладывает фильтр на строки, возвращаемые запросом. И наоборот, инструкция having накладывает фильтр на строки, возвращаемые запросом, но не влияет на вычисление агрегатной функции. Примеры см. в разделе “group by и having” на стр. 576.

order by
Сортирует набор результатов по столбцам. В Transact-SQL в инструкции order by можно указывать элементы, которые не перечислены в списке выборки. В инструкции order by можно указывать имена и заголовки (псевдонимы) столбцов, выражения, а также позиции элементов в **списке выборки** (*номер_в_списке_выборки*). Если в этой инструкции указан номер в списке выборки, столбцы, по которым нужно упорядочить набор результатов, должны быть включены в список выборки. Кроме того, в списке выборки в этом случае нельзя указывать символ “*”.

asc
Сортирует результаты по возрастанию (этот порядок сортировки используется по умолчанию).

desc
Сортирует результаты по убыванию.

compute
Используется с функциями агрегирования строк (sum, avg, min, max и count) для формирования итоговых значений по группам. Итоговые результаты отображаются в результирующем наборе запроса как дополнительные строки, что позволяет вывести с помощью одной команды как детальные, так и итоговые данные.

Вместе с инструкцией compute нельзя использовать инструкцию select into.

Если указана инструкция compute by, необходимо также указать инструкцию order by. Столбцы, указанные в инструкции compute by, должны быть подмножеством столбцов, указанных в инструкции order by (в частности, эти два набора столбцов могут в точности совпадать). Кроме того, в этих двух инструкциях столбцы должны идти в одном и том же порядке и начинаться с одного и того же выражения, а столбцы, которые были смежными в одной инструкции, должны остаться смежными и в другой.

Например, для инструкции `order by a, b, c` возможны следующие варианты инструкции compute by:

```
compute by a, b, c
compute by a, b
compute by a
```

Ключевое слово compute можно использовать без ключевого слова by для создания общих итогов, подсчета общего количества строк и т. п. В этом случае инструкцию order by указывать необязательно. Подробности и примеры см. в разделе “compute” на стр. 327.

Если службы Component Integration Services включены, инструкцию compute нельзя использовать для удаленных серверов.

for {read only | update}

Указывает, является ли набор результатов курсора обновляемым или доступным только для чтения. Этот параметр можно использовать только в хранимой процедуре, и только если запрос для курсора определен в этой процедуре. В этом случае в процедуре допустим только оператор select. Именно оператор select (а не оператор declare cursor) определяет режим курсора (for read only или for update). Преимущество этого метода объявления курсоров заключается в том, что при извлечении строк накладываются блокировки на уровне страницы.

Для оператора select, содержащегося в хранимой процедуре и не определяющего курсор, Adaptive Server игнорирует параметр for read only | update. Дополнительную информацию об объявлении курсоров с помощью хранимых процедур см. в документации по языку Embedded SQL™, а об обновляемых курсорах и курсорах только для чтения – в книге *Transact-SQL User’s Guide*.

of список_имен_столбцов

Столбцы из результирующего набора курсора, указанные после ключевого слова for update, можно обновлять.

at isolation

Указывает уровень изоляции запроса (0, 1, 2 или 3). Если не указать эту инструкцию, в запросе будет использоваться уровень изоляции сеанса, в котором он выполняется (по умолчанию – уровень изоляции 1). Инструкция at isolation допустима только в запросах, не являющихся частью другой команды (см. ниже), и в команде declare cursor. При использовании инструкции at isolation в следующих конструкциях СУБД Adaptive Server выдает сообщение о синтаксической ошибке:

- В запросе с инструкцией into.
- В подзапросе.
- В запросе, являющемся частью команды create view.
- В запросе, являющемся частью команды insert.
- В запросе с инструкцией for browse.

Если запрос содержит оператор union, инструкция at isolation должна быть указана после последнего запроса select, участвующего в объединении. Если в запросе, в котором указан уровень изоляции at isolation read uncommitted, также указан параметр holdlock, noholdlock или shared, Adaptive Server выдаст предупреждение и проигнорирует инструкцию at isolation. Для других уровней изоляции параметр holdlock имеет приоритет над инструкцией at isolation. Дополнительную информацию об уровнях изоляции см. в книге *Transact-SQL User's Guide*.

Если службы Component Integration Services включены, параметр at isolation нельзя использовать для удаленных серверов.

read uncommitted | 0

Указывает для запроса уровень изоляции 0.

read committed | 1

Указывает для запроса уровень изоляции 1.

repeatable read | 2

Указывает для запроса уровень изоляции 2.

serializable | 3

Указывает для запроса уровень изоляции 3.

for browse

Этот параметр должен быть указан в конце команды SQL, отправленной на сервер Adaptive Server из приложения, использующего библиотеку DB-Library. Дополнительную информацию см. в книге *Open Client DB-Library Reference Manual*.

plan "абстрактный план"

Указывает абстрактный план, который будет использоваться для оптимизации запроса. Это может быть полный или частичный план, написанный на языке абстрактных планов. Дополнительную информацию см. в главе 30, “Руководство по написанию абстрактных планов”, книги *Руководство по настройке производительности*.

Примеры

Пример 1. Выбирает все строки и столбцы из таблицы publishers:

```
select * from publishers
```

```
pub_id pub_name                    city                 state
------ --------------------------- -------------------- -----
0736    New Age Books              Boston               MA
0877    Binnet & Hardley           Washington           DC
1389    Algodata Infosystems       Berkeley             CA
```

Пример 2. Выбирает все строки из определенных столбцов таблицы publishers:

```
select pub_id, pub_name, city, state from publishers
```

Пример 3. Выбирает все строки из определенных столбцов таблицы publishers, подставляя вместо имени одного столбца псевдоним и добавляя строку к набору результатов:

```
select "The publisher's name is",
Publisher = pub_name, pub_id
from publishers
```

```
                      Publisher                      pub_id
---------------------- ------------------------------ ------
The publisher’s name is New Age Books                0736
The publisher’s name is Binnet & Hardley             0877
The publisher’s name is Algodata Infosystems         1389
```

Пример 4. Выбирает все строки из определенных столбцов таблицы titles, подставляя вместо имен столбцов псевдонимы:

```
select type as Type, price as Price
from titles
```

Пример 5. Указывает схему блокировки и интервал между резервными страницами для команды select into:

```
select title_id, title, price
into bus_titles
lock datarows with reservepagegap = 10
from titles
where type = "business"
```

Пример 6. Выбирает только строки, не блокированные в монопольном режиме. Если какой-то другой пользователь установил монопольную блокировку на строку, удовлетворяющую условию запроса, эта строка не возвращается:

```
select title, price
from titles readpast
    where type = "news"
    and price between $20 and $30
```

Пример 7. Выбирает определенные столбцы и строки, помещая результаты во временную таблицу #advance_rpt:

```
select pub_id, total = sum (total_sales)
    into #advance_rpt
from titles
where advance < $10000
    and total_sales is not null
group by pub_id
having count(*) > 1
```

Пример 8. Выполняет конкатенацию двух столбцов и помещает результаты во временную таблицу #tempnames:

```
select "Author_name" = au_fname + " " + au_lname
    into #tempnames
    from authors
```

Пример 9. Выбирает определенные столбцы и строки, возвращает результаты, упорядоченные по столбцу type (от наибольшего к наименьшему), и вычисляет итоговые значения:

```
select type, price, advance from titles
order by type desc
compute avg(price), sum(advance) by type
compute sum(price), sum(advance)
```

Пример 10. Выбирает определенные столбцы и строки и вычисляет итоги для столбцов price и advance:

```
select type, price, advance from titles compute sum(price), sum(advance)
```

Пример 11. Создает таблицу coffeetabletitles, имеющую ту же структуру, что и таблица titles, и заносит в нее строки этой таблицы, соответствующие книгам дороже $20:

```
select * into coffeetabletitles from titles
where price > $20
```

Пример 12. Создает таблицу newtitles, имеющую ту же структуру, что и таблица titles, но не содержащую данных:

```
select * into newtitles from titles
where 1 = 0
```

Пример 13. Заносит в существующую таблицу authors строки с книгами дороже $20:

```
select * into authors from titles
where price > $20
```

Пример 14. Дает подсказку оптимизатору:

```
select title_id, title
    from titles (index title_id_ind prefetch 16)
    where title_id like "BU%"
```

Пример 15. Выбирает столбец IDENTITY из таблиц sales_east и sales_west, используя ключевое слово syb_identity:

```
select sales_east.syb_identity,
sales_west.syb_identity
from sales_east, sales_west
```

Пример 16. Создает таблицу newtitles, являющуюся копией таблицы titles, плюс столбец IDENTITY:

```
select *, row_id = identity(10)
into newtitles from titles
```

Пример 17. Указывает уровень изоляции транзакции для запроса.

```
select pub_id, pub_name
from publishers
at isolation read uncommitted
```

Пример 18. Выполняет запрос к таблице titles, используя уровень изоляции repeatable read. Ни один другой пользователь не может изменять или удалять выбранные запросом строки, пока транзакция не завершится:

```
begin tran
select type, avg(price)
    from titles
    group by type
at isolation repeatable read
```

Пример 19. Дает подсказку оптимизатору о степени параллелизма для запроса:

```
select ord_num from salesdetail
    (index salesdetail parallel 3)
```

Пример 20. Соединяет таблицы titleauthor и titles по столбцу title_id. В набор результатов включаются только строки, для которых значение столбца price больше 15:

```
select au_id, titles.title_id, title, price
from titleauthor inner join titles
on titleauthor.title_id = titles.title_id
and price > 15
```

Пример 21. Этот набор результатов содержит всех авторов из таблицы authors. Для авторов, которые не живут в одном городе со своим издателем, столбец pub_name равен NULL. Только для авторов, которые живут в одном городе со своим издателем (Cheryl Carson и Abraham Bennet), столбец pub_name не равен NULL:

```
select au_fname, au_lname, pub_name
from authors left join publishers
on authors.city = publishers.city
```

Пример 22. Создает новую таблицу (newtable) на основе существующей (oldtable) и задает интервал между значениями столбца IDENTITY:

```
select identity into newtable
with identity_gap = 20
from oldtable
```

Дополнительную информацию см. в разделе “Managing Identity Gaps in tables” главы 7, “Creating Databases and Tables”, книги *Transact-SQL User’s Guide*.

Использование

- Ключевые слова в команде select, как и во всех других командах, должны идти в порядке, указанном в синтаксисе.
- Максимальное количество выражений в команде select – 4096.
- Чтобы обеспечить совместимость с другими реализациями SQL, после ключевого слова select можно указать ключевое слово all (хотя это делать необязательно, так как all – режим по умолчанию). В этом контексте ключевое слово all противоположно distinct. Оно включает в результирующий набор все найденные строки, в том числе и повторяющиеся.
- Во всех командах, кроме create table, create view и select into, заголовки столбцов могут содержать любые символы, в том числе пробелы и ключевые слова Adaptive Server (при условии, что они заключены в кавычки). Если заголовок не заключен в кавычки, он должен соответствовать правилам именования идентификаторов.
- Символьная строка, указанная с ключевым словом like, не может быть длиннее 255 байтов.

- Команду select...for browse нельзя выполнять над таблицами, содержащими более 255 столбцов.
- Заголовки столбцов и псевдонимы таблиц в командах create table, create view и select into должны соответствовать правилам именования идентификаторов.
- Если команда select используется для вставки данных, то может случиться так, что исходная таблица содержит значения NULL, а в принимающей таблице значения NULL не допускаются. В этом случае необходимо указать значение, которое будет подставлено вместо значений NULL исходной таблицы. Например, для вставки данных в таблицу advances, которая не допускает значения NULL, в приведенном ниже примере значения NULL заменяются на “0”:

  ```
  insert advances
  select pub_id, isnull(advance, 0) from titles
  ```

  Если бы не было этой функции isnull, эта команда вставила бы все строки со значениями, отличными от NULL, в таблицу advances и выдала бы сообщения об ошибках для всех строк таблицы titles, для которых столбец advance равен NULL.

  Если такую замену не сделать, нельзя будет вставить данные со значениями NULL в столбцы с ограничением NOT NULL.

  Две таблицы могут иметь идентичную структуру, но отличаться тем, что для какого-либо столбца в одной таблице разрешены значения NULL, а для соответствующего столбца в другой – не разрешены. Чтобы узнать, какие столбцы допускают значения NULL, выполните процедуру sp_help.
- Длина по умолчанию для данных типа text или image, возвращаемых командой select, равна 32 КБ. Это значение можно изменить с помощью команды set textsize. Значение для текущего сеанса хранится в глобальной переменной *@@textsize*. Некоторые клиентские программы могут выполнять команду set textsize при подключении к серверу Adaptive Server.
- Данные с удаленных серверов Adaptive Server можно извлечь с помощью вызовов удаленных процедур. Подробности см. в описании команд create procedure и execute.
- Команда select, используемая в определении курсора (команде declare cursor), должна содержать инструкцию from , но не может содержать инструкций compute, for browse и into. Если команда select содержит какую-либо из следующих конструкций, курсор считается необновляемым и доступным только для чтения:

- distinct, параметр
- group by, инструкция
- Агрегатные функции
- union, оператор

Если в хранимой процедуре объявляется курсор с помощью команды select , содержащей инструкцию order by, то этот курсор считается доступным только для чтения. Через курсор, определенный командой select, содержащей соединение двух и более таблиц, нельзя удалить строку (даже если курсор считается обновляемым). Подробности см. в описании команды declare cursor.

- Если команда select, присваивающая значение переменной, возвращает более одной строки, переменной присваивается последнее возвращенное значение. Например:

```
declare @x varchar(40)
select @x = pub_name from publishers
print @x
(3 rows affected)
Algodata Infosystems
```

Использование синтаксиса ANSI-соединений

- Перед тем как приступить к написанию запросов с использованием синтаксиса внутренних и внешних ANSI-соединений, прочитайте раздел “Outer Joins” главы 4, “Joins: Retrieving Data From Several Tables”, книги *Transact-SQL User’s Guide*.

Использование команды *select into*

- select into является двухшаговой операцией. На первом шаге создается новая таблица, а на втором в нее вставляются указанные строки.

Примечание. С помощью команды select into можно вставлять строки и в существующую таблицу.

Поскольку операции вставки строк с помощью команды select into не записываются в журнал, команды select into нельзя выполнять внутри пользовательских транзакций, даже если параметр базы данных ddl in tran равен true. Однако выделение страниц при выполнении операций select into записывается в журнал, поэтому операции select into могут заполнить журнал транзакций.

Если в команде select into происходит сбой после создания новой таблицы, то таблица *не* удаляется автоматически и ее первая страница данных не освобождается. Это означает, что все строки, вставленные на первую страницу до ошибки, остаются на этой странице. Проверьте значение глобальной переменной *@@error* после выполнения команды select into, чтобы убедиться в отсутствии ошибок. Удалите новую таблицу с помощью команды drop table, а затем повторно выполните команду select into.

- Имя новой таблицы должно быть уникально в базе данных и соответствовать правилам именования идентификаторов. С помощью команды select into можно вставить строки во временные таблицы (см. примеры 7, 8 и 11).
- Никакие правила, ограничения или значения по умолчанию, связанные с исходной таблицей, не переносятся на новую таблицу. Для привязки правил или значений по умолчанию к новой таблице нужно выполнить системные процедуры sp_bindrule и sp_bindefault.
- Команда select into не копирует в новую таблицу значение параметра max_rows_per_page для исходной таблицы. В новой таблице значение параметра max_rows_per_page будет равно 0. Чтобы присвоить ему значение, выполните системную процедуру sp_chgattribute.
- Чтобы с помощью команды select into можно было вставить строки в постоянную таблицу, параметру select into/bulkcopy/pllsort нужно присвоить значение true (с помощью системной процедуры sp_dboption). Если же команда select into используется для вставки во временную таблицу, параметр select into/bulkcopy/pllsort не требуется устанавливать в true, поскольку временная база данных никогда не восстанавливается.

  После выполнения команды select into необходимо сделать полную резервную копию базы данных, прежде чем можно будет использовать команду dump transaction. Для операций select into записывается в журнал только выделение страниц, но не изменение строк данных. Следовательно, изменения нельзя восстановить по журналам транзакций. В такой ситуации при запуске команды dump transaction будет выдано сообщение об ошибке, указывающее, что вместо него нужно выполнить команду dump database.

  По умолчанию параметр select into/bulkcopy/pllsort в созданных базах данных устанавливается в false. Можно изменить это значение по умолчанию, установив этот параметр в true в базе данных model.
- Команда select into работает медленнее, если в это же время выполняется команда dump database.

- С помощью команды select into можно скопировать структуру таблицы, не занося в новую таблицу никаких данных. Для этого нужно задать ложное условие в инструкции where (см. пример 12).
- Каждому столбцу в списке выборки, содержащему агрегатную функцию или выражение, нужно назначить заголовок. Это относится ко всем константам, арифметическим и символьным выражениям, встроенным функциям и элементам, образуемым путем конкатенации других элементов в списке выборки. Заголовок столбца должен быть допустимым идентификатором или должен быть заключен в кавычки (см. примеры 7 и 8).
- Для столбцов таблицы, создаваемой командой select into, типы данных и допустимость значений NULL назначаются неявно, например:

  ```
  select x = getdate() into mytable
  ```

  В этом случае столбец новой таблицы не будет допускать значений NULL, независимо от значения параметра allow nulls by default. Это зависит от того, в каком контексте используется команда select и какие еще команды есть в запросе.

  С помощью синтаксиса convert можно явно указать тип данных и допустимость значений NULL для результирующих столбцов, а не принимать настройки по умолчанию.

  Можно применить к getdate функцию, результатом которой может быть NULL, например:

  ```
  select x = nullif(getdate(), "1/1/1900") into
  mytable
  ```

  или использовать функцию convert:

  ```
  select x = convert(datetime null, getdate()) into
  mytable
  ```

- Команду select into нельзя использовать в пользовательских транзакциях и вместе с инструкцией compute.
- Чтобы выбрать столбец IDENTITY в результирующую таблицу, нужно включить имя этого столбца (или ключевое слово syb_identity) в *список_столбцов* команды select. Новый столбец подчиняется следующим правилам:
  - Если столбец IDENTITY выбирается несколько раз, то в новой таблице он будет определен как NOT NULL. Этот столбец не унаследует свойство IDENTITY.

  - Если столбец IDENTITY указан в списке выборки в составе выражения, результирующий столбец также не наследует свойство IDENTITY. Новый столбец будет допускать значения NULL, если какой-либо из столбцов в выражении допускает значения NULL; в противном случае он создается с ограничением NOT NULL.
  - Если команда select содержит инструкцию group by или агрегатную функцию, результирующий столбец не унаследует свойство IDENTITY. Столбцы, содержащие агрегатную функцию над столбцом IDENTITY, будут допускать значения NULL, остальные – нет.
  - Столбец IDENTITY, который выбирается в таблицу с объединением (union) или соединением (join), не сохраняет свойство IDENTITY. Если таблица содержит объединение столбца IDENTITY и столбца, допускающего значения NULL, новый столбец тоже будет допускать значения NULL. В противном случае он будет определен как NOT NULL.

- С помощью команды select into нельзя создавать таблицу с несколькими столбцами IDENTITY. Если в команде select указан как существующий столбец IDENTITY, так и новая спецификация IDENTITY в виде *имя_столбца* = identity*(точность)*, эта команда не будет выполнена и возвратит ошибку.
- Если службы Component Integration Services включены и таблица, указанная после ключевого слова into, хранится на сервере Adaptive Server, то сервер использует подпрограммы массового копирования для копирования данных в новую таблицу. Перед выполнением команды select into с удаленными таблицами задайте параметру базы данных select into/bulkcopy значение true.
- Дополнительную информацию о команде Embedded SQL select into *список_переменных_базового_языка* см. в книге *Open Client Embedded SQL Reference Manual*.

Изменение допустимости значений NULL для результирующего столбца в команде *select...into*

- С помощью команды convert можно разрешить или запретить значения NULL в столбце, в который выбираются данные. Например, следующая команда выбирает данные из таблицы titles в таблицу temp_titles, при этом для столбца total_sales устанавливается ограничение not null (хотя для соответствующего столбца в исходной таблице этого ограничения не было):

```
select title, convert (char(100) not null,
total_sales)
total_sales
into #tempsales
from titles
```

Указание схемы блокировки с помощью *select...into*

- Параметр lock команды select...into позволяет указать схему блокировки для таблицы, создаваемой этой командой. Если не указать схему блокировки, применяется схема блокировки по умолчанию, заданная параметром конфигурации lock scheme.
- При использовании параметра lock также можно указать свойства управления пространством max_rows_per_page, exp_row_size и reservepagegap.

  Свойства управления пространством для таблицы, созданной с помощью команды select into, можно изменить с помощью системной процедуры sp_chgattribute.

Использование *index*, *prefetch* и *lru | mru*

- Параметры index, prefetch и lru | mru позволяют указать индекс, стратегии кэширования и ввода-вывода для выполнения запроса. Эти параметры переопределяют значения, выбранные оптимизатором Adaptive Server. Их следует использовать очень осторожно, а также всегда проверять их влияние на производительность с помощью команды set statistics io on. Дополнительную информацию об этих параметрах см. в книге *Руководство по настройке производительности*.

Использование параметра *parallel*

- Параметр parallel уменьшает число рабочих потоков, которые оптимизатор Adaptive Server может использовать для параллельной обработки. Значение *степени_параллелизма* не может превышать значение max parallel degree. Если значение, указанное в параметре parallel, больше значения max parallel degree, этот параметр будет проигнорирован оптимизатором.
- Когда несколько рабочих процессов объединяют свои результаты, порядок строк, возвращаемых Adaptive Server, не определен и может быть разным при выполнении одной и той же команды. Чтобы получить строки из секционированной таблицы в согласованном порядке, используйте инструкцию order by или отмените параллельное выполнение запроса, указав параметр parallel 1 в инструкции from запроса.
- Инструкция from с параметром parallel игнорируется в следующих случаях:

- Команда select используется для обновления или вставки.
- Инструкция from используется в определении курсора.
- Параметр parallel используется в инструкции from во внутреннем блоке подзапроса.
- Команда select создает представление.
- Таблица является внутренней во внешнем соединении.
- В запросе используются функции min или max над данными таблицы и указан индекс.
- Указан несекционированный кластерный индекс, или только с его помощью можно выполнить запрос параллельно.
- В запросе указана инструкция exists для таблицы.
- Значение параметра конфигурации max scan parallel degree равно 1, и в запросе указан индекс.
- Запрос покрывает некластерный индекс. Дополнительную информацию о покрытии запросов индексами см. в главе 9, "Как работают индексы", книги *Руководство по настройке производительности*.
- Таблица является системной или виртуальной.
- Запрос обрабатывается по стратегии OR. Описание стратегии OR см. в книге *Руководство по настройке производительности*.
- Запрос выдает пользователю большое количество строк.

Использование параметра *readpast*

- Параметр readpast позволяет команде select обращаться к указанной таблице, не ожидая снятия несовместимых блокировок, удерживаемых другими задачами. Параметр readpast можно указывать только в запросах к таблицам с блокировкой только данных.
- Если параметр readpast указан для таблицы с блокировкой всех страниц, он игнорируется. Команда выполняется с уровнем изоляции, указанным для этой команды или для сеанса. Если используется уровень изоляции 0, выполняется "грязное" чтение. В этом случае команда возвращает значения из заблокированных строк и не ожидает снятия блокировки. Если же уровень изоляции – 1 или 3, команда ожидает снятия несовместимых блокировок со страниц, которые она должна прочитать.

- Комбинации уровней изоляции, заданных для сеанса, и значений параметра readpast для таблицы, заданных в команде select, описаны в таблице 7-32.

Таблица 7-32. Комбинации уровней изоляции для сеанса и значений параметра readpast

Уровень изоляции сеанса	Результат
0, read uncommitted (“грязные” чтения)	Параметр readpast игнорируется, и пользователю возвращаются строки, содержащие еще незафиксированные данные. Также выводится соответствующее предупреждение.
1, read committed	Строки или страницы с несовместимыми блокировками не выбираются запросом; читаемые строки или страницы не блокируются. Запрос с ключевым словом readpast может возвратить одинаковые строки, причем добавление инструкции distinct не решит проблему. Чтобы избежать этого, вместе с ключевым словом readpast нужно указать *не только* инструкцию distinct, но и инструкцию group by.
2, repeatable read	Строки или страницы с несовместимыми блокировками не выбираются запросом; для всех считываемых строк или страниц удерживаются разделяемые блокировки до окончания команды или транзакции; блокируются все страницы, считываемые командой, до завершения транзакции.
3, serializable	Параметр readpast игнорируется, команда выполняется с уровнем изоляции 3. Выполнение команды прерывается, если она встречает строку или страницу с несовместимыми блокировками.

- Команда select с параметром readpast не будет выполнена и выдаст сообщение об ошибке, если она содержит:
  - инструкцию at isolation, задающую уровень изоляции 0 (read uncommitted);
  - инструкцию at isolation, задающую уровень изоляции 3 (serializable);
  - ключевое слово holdlock для той же таблицы.
- Если в запросе select, содержащем параметр readpast, также указан параметр at isolation 2 (или at isolation repeatable read), то разделяемые блокировки таблиц, для которых указано ключевое слово readpast, удерживаются до завершения команды или транзакции.
- Если команда select с параметром readpast обнаруживает текстовый столбец, заблокированный несовместимой блокировкой, соответствующая строка будет извлечена, но значение этого текстового столбца будет равно null. В этом случае нельзя будет понять, был ли текстовый столбец равен NULL в исходной таблице, или значение NULL было возвращено из-за того, что столбец блокирован.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем стандарта.

Тем не менее, в Transact-SQL существуют следующие расширения этой команды:

- инструкция into для создания новой таблицы;
- инструкции lock;
- инструкции compute;
- глобальные и локальные переменные;
- инструкции index , prefetch, parallel и lru | mru;
- ключевые слова holdlock, noholdlock и shared;
- синтаксис "*заголовок_столбца* = *имя_столбца*";
- полные имена таблиц и столбцов;
- select в инструкции for browse
- использование в списке выборки столбцов, которых нет в списке group by и к которым не применяются агрегатные функции;
- параметр at isolation repeatable read | 2.

Полномочия

По умолчанию команду select разрешено выполнять владельцу таблицы или представления, который может предоставить это право другим пользователям.

См. также

Команды compute, create index, create trigger, delete, group by и having, insert, order by, set, union, update, where

Функции avg, count, isnull, max, min,sum

Системные процедуры sp_cachestrategy, sp_chgattribute, sp_dboption

set

Описание

Задает параметры обработки запросов, действующие в течение рабочего сеанса пользователя, а также некоторые параметры внутри триггеров или хранимых процедур.

Синтаксис

set ansinull {on | off}

set ansi_permissions {on | off}

set arithabort [arith_overflow | numeric_truncation]
{on | off}

set arithignore [arith_overflow] {on | off}

set {chained, close on endtran, nocount, noexec, parseonly, procid, self_recursion, showplan, sort_resources} {on | off}

set char_convert {off | on [with {error | no_error}] |
charset [with {error | no_error}]}

set cis_rpc_handling {on | off}

set [clientname *имя_клиента* | clienthostname
имя_узла | clientapplname *имя_приложения*]

set cursor rows *количество* for *имя_курсора*

set {datefirst *число*, dateformat *формат*,
language *язык*}

set explicit_transaction_required [true | false]

set fipsflagger {on | off}

set flushmessage {on | off}

set forceplan {on | off}

set identity_insert [*база_данных*.[*владелец*.]]*имя_таблицы*
{on | off}

set jtc {on | off}

set lock { wait [количество_секунд] | nowait }

set offsets {select, from, order, compute, table,
procedure, statement, param, execute} {on | off}

set parallel_degree *число*

set plan {dump | load } [*имя_группы*] {on | off}

set plan exists check {on | off}

set plan replace {on | off}

set prefetch [on|off]

set process_limit_action {abort | quiet | warning}

set proxy login_name

```
set quoted_identifier {on | off}

set role {"sa_role" | "sso_role" | "oper_role" |
        имя_роли [with passwd "пароль"]} {on | off}

set {rowcount количество, textsize количество}

set scan_parallel_degree количество

set session authorization login_name

set sort_merge {on | off}

set statistics {io, subquerycache, time} {on | off}

set statistics simulate { on | off }

set strict_dtm_enforcement {on | off}

set string_rtruncation {on | off}

set table count количество

set textsize {количество}

set transaction isolation level {
        [ read uncommitted | 0 ] |
        [ read committed | 1 ] |
        [ repeatable read | 2 ]|
        [ serializable | 3 ] }

set transactional_rpc {on | off}
```

Параметры

ansinull

Определяет, обрабатываются ли операнды со значением NULL в агрегатных функциях в соответствии со стандартом SQL92. Если параметр set ansinull равен on, Adaptive Server выдает предупреждение, когда агрегатная функция исключает операнд со значением NULL из вычисления. Этот параметр не влияет на обработку Adaptive Server значений NULL при сравнении на равенство (=) или неравенство (!=).

Например, если параметр set ansinull имеет значение off (значение по умолчанию), то результатом следующего запроса к таблице titles:

```
select max(total_sales) from titles
```

будет:

```
-----------
      22246
```

Если же этот параметр равен on, этот запрос возвратит то же значение, а также сообщение об ошибке, поскольку столбец total_sales содержит значения NULL:

```
-----------
      22246
Warning - null value eliminated in set function
```

Это сообщение указывает, что некоторые строки в этой таблице содержат значение NULL в столбце total_sales, то есть нет полной информации о суммарных продажах по всем книгам. Однако из доступных данных возвращенное значение является наибольшим.

ansi_permissions

Определяет, проверяется ли соблюдение требований SQL92 по полномочиям для команд delete и update. Значение по умолчанию – off. Требования по полномочиям приведены в таблице 7-33:

Таблица 7-33. Полномочия, необходимые для операций обновления и удаления данных

Команда	Полномочия, необходимые при параметре set ansi_permissions, равном off	Полномочия, необходимые при параметре set ansi_permissions, равном on
update	• Полномочия update для столбцов, чьи значения устанавливаются этой командой	• Полномочия update для столбцов, чьи значения устанавливаются этой командой • Полномочия select для всех столбцов, перечисленных в инструкции where • Полномочия select для всех столбцов, указанных в правой части инструкции set
delete	• Полномочия delete для таблицы	• Полномочия delete для таблицы. • Полномочия select для всех столбцов, перечисленных в инструкции where

arithabort

Определяет поведение Adaptive Server при возникновении арифметических ошибок. Параметры arithabort arith_overflow и arithabort numeric_truncation обрабатывают разные типы арифметических ошибок. Эти параметры можно устанавливать как отдельно друг от друга, так и оба сразу с помощью оператора set arithabort on или set arithabort off.

- Параметр arithabort arith_overflow определяет поведение Adaptive Server при ошибке, связанной с делением на ноль или с потерей точности при явном или неявном преобразовании типа данных. Ошибки этого типа серьезны. Если задан параметр arithabort arith_overflow on (значение по умолчанию), то происходит откат всей транзакции, в которой встречается такая ошибка. Если эта ошибка происходит в пакете, который не содержит транзакций, то при настройке arithabort arith_overflow on не происходит откат предыдущих команд пакета; СУБД Adaptive Server не выполняет все остальные команды пакета, следующие за ошибочной командой.

Если же задано arithabort arith_overflow off, Adaptive Server отменяет команду, вызвавшую ошибку, но продолжает обработку других команд в транзакции или пакете.

- Параметр arithabort numeric_truncation определяет поведение Adaptive Server при потере масштаба числовым типом данных с точным представлением, возникающей при неявном преобразовании типов данных. (Если потеря масштаба происходит при явном преобразовании типов, из результата выбрасываются потерянные цифры, а предупреждение не выдается). Если этот параметр равен on (значение по умолчанию), Adaptive Server отменяет команду, вызвавшую ошибку, но продолжает обработку других команд в транзакции или пакете. Если же этот параметр равен off, Adaptive Server выбрасывает из результатов запроса потерянные цифры и продолжает обработку.

arithignore arith_overflow

Определяет, выводит ли Adaptive Server сообщение после ошибки, связанной с делением на ноль или потерей точности. По умолчанию параметр arithignore равен off. В этом случае Adaptive Server выводит предупреждающее сообщение, если при выполнении запроса возникает числовое переполнение. Чтобы Adaptive Server игнорировал ошибки переполнения, установите параметр arithignore в on. Можно опустить необязательное ключевое слово arith_overflow.

chained

Начинает транзакцию в начале сеанса и после завершения другой транзакции, сразу перед первой командой, выполняющей поиск или изменение данных. В связанном режиме транзакций сервер Adaptive Server неявно выполняет команду begin transaction перед командами delete, fetch, insert, open, select и update. Команду set chained нельзя выполнять в транзакции.

char_convert

Включает и отключает преобразование набора символов между Adaptive Server и клиентом. Если клиент использует библиотеку Open Client DB-Library версии 4.6 или выше, а его набор символов отличается от набора символов на клиенте, преобразование включается во время соединения с сервером, а его входной набор символов выбирается по умолчанию согласно набору символов, используемому клиентом. Можно также явно указать, какой набор символов будет являться входным для преобразования, с помощью команды set char_convert набор_символов.

charset – это идентификатор набора символов либо имя из таблицы syscharsets со значением столбца type меньше 2000.

Настройка set char_convert off отключает преобразование, в результате чего символы не меняются при отправке и получении. Параметр set char_convert on включает преобразование, если оно в настоящий момент отключено. Если преобразование набора символов не было включено во время подключения к серверу или с помощью команды set char_convert, то команда set char_convert on выдает сообщение об ошибке.

Если было включено преобразование набора символов с помощью команды set char_convert набор_символов, но Adaptive Server не может выполнить это преобразование, то состояние преобразования останется тем же, каким было до запроса. Например, если до выполнения команды set char_convert набор_символов преобразование было отключено, оно останется отключенным при ошибке в запросе.

Если в команде set char_convert указан параметр with no_error, сервер Adaptive Server не будет уведомлять приложение о том, что набор символов сервера не может быть преобразован в набор символов клиента. Выдача сообщений об ошибках изначально включена при подключении клиента к Adaptive Server. Чтобы не получать сообщения об ошибках, ее можно отключить для всех сеансов с помощью команды set char_convert {on | charset} with no_error. Чтобы включить сообщения об ошибках во время сеанса, выполните команду set char_convert {on | charset} with error.

Независимо от того, включена ли выдача сообщений об ошибках, байты, которые нельзя преобразовать, заменяются вопросительными знаками ASCII (?).

Дополнительную информацию об обработке ошибок при преобразовании символов см. в книге *Руководство по системному администрированию*.

cis_rpc_handling
: Определяет, обрабатываются ли по умолчанию исходящие вызовы удаленных процедур службами Component Integration Services.

clientapplname
: Назначает приложению индивидуальное имя. Это позволяет различать клиентов в системе, где множество клиентов подключаются к Adaptive Server, используя одно и то же имя приложения. Новое имя приложения будет также занесено в таблицу sysprocesses.

clienthostname
: Назначает узлу индивидуальное имя. Это позволяет различать клиентов в системе, где множество клиентов подключаются к Adaptive Server, используя одно и то же имя узла. Новое имя хоста будет также занесено в таблицу sysprocesses.

clientname

Назначает клиенту индивидуальное имя. Это позволяет различать клиентов в системе, где множество клиентов подключаются к Adaptive Server, используя одно и то же имя клиента. Новое имя пользователя будет также занесено в таблицу sysprocesses.

close on endtran

Указывает серверу Adaptive Server закрыть все открытые в транзакции курсоры в конце этой транзакции. Транзакция завершается командой commit или rollback. Следует отметить, что будут закрыты только курсоры, объявленные в блоке, в котором был задан этот параметр (хранимой процедуре, триггере и т. п.). Дополнительную информацию об областях действия курсоров см. в книге *Transact-SQL User's Guide*.

Дополнительную информацию об опробованной конфигурации см. в книге *Руководство по системному администрированию*.

cursor rows

Указывает, что будет возвращаться количество строк, равное значению параметра *количество*, при каждом запросе на извлечение данных через курсор, поступающем от клиентского приложения. Значение параметра *количество* может быть числовой константой без десятичной точки или локальной переменной типа integer. Если в качестве *количества* было задано неположительное значение, этот параметр устанавливается в 1. Параметр cursor rows можно установить для курсора независимо от того, закрыт он или открыт. Однако этот параметр не действует, если команда fetch содержит инструкцию into. Параметр *имя_курсора* указывает курсор, для которого задается количество возвращаемых строк.

datefirst

Присваивает первому дню недели номер (от 1 до 7). Для языка us_english значение по умолчанию – 1 (воскресенье).

dateformat

Задает порядок, в котором идут части даты *(месяц, день и год)* для ввода значения типа данных datetime или smalldatetime. Допустимые значения: *mdy* (месяц/день/год), *dmy* (день/месяц/год), *ymd* (год/месяц/день), *ydm* (год/день/месяц), *myd* (месяц/год/день) и *dym* (день/год/месяц). Для языка us_english значение по умолчанию – *mdy*.

explicit_transaction_required

Если этот параметр установлен в true, нельзя будет начать неявную транзакцию, а также отправить вызов удаленной процедуры на удаленный сервер вне транзакции.

Все остальные команды будут выполнены успешно.

fipsflagger

Определяет, будет ли выдаваться предупреждение при использовании расширения базового уровня SQL92, доступное в Transact-SQL. По умолчанию Adaptive Server не выводит предупреждения при использовании нестандартного SQL. Этот параметр не отключает расширений SQL. Обработка будет нормально завершена при вводе команды, не соответствующей стандарту ANSI SQL.

flushmessage

Определяет, когда Adaptive Server будет выдавать сообщения пользователю. По умолчанию сообщения хранятся в буфере, пока сформировавший их запрос не завершится или буфер не переполнится. Если выполнить команду set flushmessage on, сообщения будут выдаваться пользователю сразу после того, как они были сформированы.

forceplan

Указывает оптимизатору запросов, что в плане запроса должен использоваться такой же порядок соединения, что и порядок таблиц в инструкции from этого запроса. Параметр forceplan обычно используется, когда оптимизатору не удается выбрать хороший план. Навязывание неправильного плана может существенно понизить скорость ввода-вывода и производительность. Дополнительную информацию см. в книге *Руководство по настройке производительности*.

identity_insert

Определяет, допустимы ли явные вставки данных в столбец IDENTITY указанной таблицы. (Обновления столбца IDENTITY никогда не допускаются.) Этот параметр может использоваться только с базовыми таблицами. Его нельзя использовать с представлениями или задавать в триггере.

Если установить параметр identity_insert *имя_таблицы* в on, то владелец таблицы, владелец базы данных или системный администратор смогут явным образом вставлять значения в столбец IDENTITY. Вставка значения в столбец IDENTITY позволяет указать начальное значение для столбца или восстановить строку, удаленную по ошибке. Если по столбцу IDENTITY не был создан уникальный индекс, Adaptive Server не проверяет уникальность вставленного значения; поэтому можно вставлять любое положительное целое число.

Команду set identity_insert *имя_таблицы* on для таблицы со столбцом IDENTITY могут выполнять (тем самым разрешая ручную вставку значений в столбец IDENTITY) владелец таблицы, владелец базы

данных или системный администратор. Однако только следующие пользователи действительно могут вставлять значения в столбец IDENTITY, если параметр identity_insert установлен в on:

- владелец таблицы;
- владелец базы данных в следующих случаях:
  - Если владелец таблицы явно предоставил ему полномочия insert на этот столбец
  - Если он выдает себя за владельца таблицы (для чего он должен выполнить команду setuser)

Если установлено значение параметра identity_insert имя_таблицы off, то явная вставка в столбцы IDENTITY запрещается. Команду set identity_insert имя_таблицы on можно выполнить с отдельной таблицей базы данных в любое время в течение сеанса.

jtc

Включает/отключает режим транзитивного замыкания соединений. Дополнительную информацию см. в книге *Руководство по настройке производительности*.

language

Официальное название языка, на котором отображаются системные сообщения. Этот язык должен быть установлен на Adaptive Server. Значение по умолчанию – us_english.

nocount

Указывает, будет ли отображаться количество строк, обработанных командой. Установка set nocount on отключает отображение количества строк, а set nocount off – включает.

noexec

Компилирует все запросы, не выполняя их. Параметр noexec часто используется с параметром showplan. После того как установлен параметр noexec on, никакие команды (в том числе другие команды set) не выполняются, пока не будет установлен параметр noexec off.

При этой установке все запросы компилируются, но не выполняются. Команда set fmtonlyon часто используется с параметром showplan для устранения неполадок. Параметр noexec on можно установить сразу после выполнения запроса. В результате последующие команды (в том числе другие команды set) не будут выполняться, пока не будет установлен параметр noexec off. Параметр set noexec можно использовать в хранимых процедурах.

lock wait
Задает период времени, в течение которого команда может ожидать получения блокировки. Когда этот период истечет, команда завершится с сообщением об ошибке.

количество_секунд
Количество секунд, которые команда будет ожидать получения блокировки. Допустимые значения – от 0 до 2147483647 (максимальное значение для типа integer).

lock nowait
Указывает, что если команда не может установить блокировку немедленно, она завершается ошибкой. Команда set lock nowait эквивалентна set lock wait 0.

offsets
Возвращает позицию указанных ключевых слов (относительно начала запроса) в операторах Transact-SQL. Список ключевых слов, отделенных друг от друга запятыми, может содержать следующие конструкции Transact-SQL: select, from, order, compute, table, procedure, statement, param и execute. Adaptive Server возвращает смещения при отсутствии ошибок. Этот параметр может использоваться только в библиотеке Open Client DB-Library.

parallel_degree
Указывает верхний предел для количества рабочих процессов, участвующих в параллельном выполнении запроса. Это число должно быть не больше значения параметра конфигурации max parallel degree (задающего количество рабочих процессов, которое может использоваться для одного запроса). Текущая настройка хранится в глобальной переменной *@@parallel_degree*.

parseonly
Проверяет синтаксис всех запросов и выдает сообщения об ошибках, не компилируя и не выполняя запрос. Параметр parseonly нельзя использовать в хранимых процедурах и триггерах.

plan
Указывает, что команда работает с абстрактным планом. Дополнительную информацию см. в главе 30, “Создание и использование абстрактных планов”, книги *Руководство по настройке производительности*.

dump
Включает и отключает сохранение абстрактных планов для текущего соединения. Если *имя_группы* не указано, планы сохраняются в группе, установленной по умолчанию (ap_stdout).

load

Включает и отключает загрузку абстрактных планов для текущего соединения. Если *имя_группы* не указано, планы загружаются из группы, установленной по умолчанию (ap_stdin).

имя_группы

Имя группы абстрактных планов, которая должна использоваться для загрузки или хранения планов.

exists check

Если этот параметр, а также параметр set plan load установлены в on, хэш-ключи для не более чем 20 запросов из группы абстрактных планов сохраняются в кэше, выделенном пользователю.

replace

Включает и отключает замену существующих абстрактных планов в режиме сохранения планов. По умолчанию замена планов отключена.

prefetch

Включает и отключает операции ввода-вывода большого объема из кэша данных.

process_limit_action

Указывает, будет ли Adaptive Server выполнять параллельные запросы, если доступных рабочих процессов недостаточно. В этом случае, если параметр process_limit_action установлен в quiet, Adaptive Server скорректирует план (без выдачи сообщения) таким образом, чтобы степень параллелизма не превышала количество доступных процессов. Если параметр process_limit_action установлен в warning, то при недостаточном количестве доступных рабочих процессов Adaptive Server выдает предупреждающее сообщение во время корректировки плана. Если же параметр process_limit_action установлен в abort, Adaptive Server прекратит выполнение запроса и выдаст поясняющее сообщение “an insufficient number of worker processes are available” (не хватает доступных рабочих процессов).

procid

Возвращает идентификационный номер хранимой процедуры в библиотеку Open Client DB-Library/C (а не пользователю) перед отправкой строк, сформированных этой хранимой процедурой.

proxy

Позволяет использовать полномочия, регистрационное имя и suid (идентификатор пользователя на сервере) пользователя, определенного *регистрационным_именем*. *регистрационное_имя* должно быть допустимым регистрационным именем из таблицы master..syslogins,

заключенным в кавычки. Чтобы вернуться к исходному регистрационному имени и идентификатору suid, выполните команду set proxy с исходным *регистрационным_именем*.

Примечание. Ни роль sa_role, ни роль sso_role не позволяют пользователю выполнить команду set proxy login_name, если у него нет на это явных полномочий. Чтобы использовать команду set proxy login_name, любой пользователь (в том числе администратор безопасности) должен обладать полномочиями, явно предоставленными администратором безопасности.

Подробности см. в разделе “Использование команд set proxy и set session authorization” на стр. 730.

quoted_identifier

Определяет, распознает ли Adaptive Server идентификаторы с разделителями. По умолчанию параметр quoted_identifier равен off и все идентификаторы должны соответствовать правилам по именованию идентификаторов. Если параметр set quoted_identifier установлен в on, имена таблиц, представлений и столбцов могут начинаться не с буквы и включать символы, которые нельзя было бы использовать, если бы этот параметр был равен off. Кроме того, имена могут быть зарезервированными словами (такие идентификаторы нужно заключить в двойные кавычки). Идентификаторы с разделителями не могут быть длиннее 28 байтов, могут распознаваться не всеми клиентскими программами, а их использование в качестве параметров системных процедур может привести к неожиданным результатам.

Если параметр quoted_identifier установлен в on, все символьные строки, заключенные в двойные кавычки, воспринимаются как идентификаторы. Символьные или битовые строки должны быть заключены в одинарные кавычки.

роль

Включает или отключает указанную роль во время текущего сеанса. Когда пользователь соединяется с сервером, все предоставленные ему системные роли включаются. Для отключения роли нужно выполнить команду set role имя_роли off , для включения – команду set role имя_роли on. sa_role, sso_role и oper_role – это системные роли. Пользователь, не зарегистрированный в текущей базе данных, в которой нет роли пользователя-гостя, не может отключить роль sa_role, так как ему не назначен идентификатор пользователя на сервере.

имя_роли
Имя определенной пользователем роли, созданной администратором по безопасности. Роли, определенные пользователем, по умолчанию отключены. Чтобы включить такую роль при соединении с сервером, пользователь или администратор по безопасности должен выполнить команду set role on.

with пароль
Указывает пароль, который активирует роль. Если для определенной пользователем роли назначен пароль, необходимо указать именно этот пароль.

rowcount
Указывает Adaptive Server прекратить выполнение команд SQL (select, insert, update или delete) после того, как было обработано заданное количество строк. *количество* может быть числовым литералом без десятичной точки или локальной переменной типа integer. Чтобы отключить этот параметр, введите:

```
set rowcount 0
```

scan_parallel_degree
Указывает максимальную степень параллелизма в сеансе для сканирований таблицы на основе хэш-функции (параллельных сканирований индексов и несекционированных таблиц). Это число должно быть не больше текущего значения параметра конфигурации max scan parallel degree. Текущее значение хранится в глобальной переменной *@@scan_parallel_degree*.

self_recursion
Определяет, могут ли триггеры вызывать сами себя (это называется *саморекурсия*). По умолчанию Adaptive Server не допускает саморекурсии триггеров. Этот параметр можно включить только для текущего сеанса клиента; его действие ограничивается областью действия триггера, в котором он установлен. Например, после того как триггер, в котором установлен параметр self_recursion on, завершается или вызывает срабатывание другого триггера, параметр снова становится равным off. Этот параметр работает только внутри триггера и не влияет на пользовательские сеансы.

session authorization
Установка этого параметра идентична команде set proxy, за тем исключением, что set session authorization соответствует стандарту SQL, а команда set proxy является расширением, появившимся в Transact-SQL.

showplan
Формирует описание плана обработки запроса. Результаты showplan используются при диагностике производительности. Установка параметра showplan в хранимой процедуре или триггере не выводит результаты. Для параллельных запросов результаты, сгенерированные в результате установки параметра showplan, также включают в себя план запроса, скорректированный на этапе выполнения (если корректировка имела место). Дополнительную информацию см. в книге *Руководство по настройке производительности*.

sort_merge
Включает и отключает использование соединений типа "сортировка-слияние" во время сеанса. Дополнительную информацию см. в книге *Руководство по настройке производительности*.

sort_resources
Формирует описание плана сортировки для команды create index. Эти результаты используются для определения того, как будет выполняться сортировка – последовательно или параллельно. Если установлен параметр sort_resouces on, сервер Adaptive Server выводит план сортировки, но не выполняет оператор create index. Дополнительную информацию см. в главе 24, "Параллельная обработка запросов", книги *Руководство по настройке производительности*.

statistics io
Выводит следующую статистическую информацию для всех таблиц, указанных в операторе:

- число обращений к таблице (число сканирований);
- количество логических чтений (обращений к страницам в памяти);
- количество физических чтений (обращений к устройству хранения базы данных).

Для каждой команды параметр statistics io отображает количество буферов, в которые была произведена запись.

Если в сервере Adaptive Server были настроены лимиты ресурсов, параметр statistics io также отображает суммарные затраты на операции ввода-вывода. Дополнительную информацию см. в главе 35, "Использование команд set statistics", книги *Руководство по настройке производительности*.

statistics subquerycache
Отображает количество удачных и неудачных обращений к кэшу и количество строк в кэше подзапроса для каждого подзапроса.

statistics time
Отображает время, затраченное сервером Adaptive Server на анализ и компиляцию каждой команды, а также время выполнения каждого шага команды statistics time. Время приводится в миллисекундах и временных импульсах (точное значение которых зависит от компьютера).

statistics simulate
Указывает, что для оптимизации запроса оптимизатор должен использовать статистику, полученную по результатам моделирования.

strict_dtm_enforcement
Определяет, распространяет ли сервер транзакции на серверы, которые не поддерживают службы координации транзакций Adaptive Server. Значение по умолчанию наследуется от значения параметра конфигурации strict dtm enforcement.

string_rtruncation
Определяет, инициирует ли Adaptive Server исключение SQLSTATE, когда команда insert или update усекает строки типа char, unichar, varchar или univarchar . Если усекаются только пробелы, исключение не инициируется. Если параметр установлен в значение по умолчанию (off), исключение SQLSTATE не инициируется и символьная строка усекается без выдачи сообщения.

table count
Задает количество таблиц, которые сервер Adaptive Server обрабатывает за один раз при оптимизации соединения. Значение, используемое по умолчанию, зависит от количества таблиц в соединении следующим образом:

Количество таблиц, участвующих в соединении	**Количество таблиц, обрабатываемых за один раз**
2 – 25	4
26 – 37	3
38 – 50	2

Допустимые значения: 0 – 8. Значение 0 задает поведение по умолчанию. Значения выше 8 по умолчанию заменяются на 8. Установка параметра table count может улучшить оптимизацию некоторых запросов с соединениями, но увеличивает время компиляции.

textsize
: Указывает максимальный размер в байтах для данных типа text и image, которые возвращаются командой select. Текущее значение хранится в глобальной переменной *@@textsize*. Чтобы вернуть параметру textsize значение по умолчанию (32 КБ), введите:

```
set textsize 0
```

Значение по умолчанию в утилите isql равно 32 КБ. Некоторые клиентские программы устанавливают другие значения по умолчанию.

transaction isolation level
: Задает уровень изоляции транзакций для сеанса. Все транзакции, выполняющиеся в момент установки этого параметра, а также запущенные после этой установки, будут использовать этот уровень изоляции.

read uncommitted | 0
: Если установлен уровень изоляции 0, никакие блокировки не применяются. Следовательно, в этом случае набор результатов, по которому происходит сканирование, может измениться во время сканирования. Если позиция сканирования потеряна из-за изменений в базовой таблице, для повторного запуска сканирования необходим уникальный индекс. При отсутствии уникального индекса сканирование может быть прекращено.

По умолчанию для сканирования (при уровне изоляции 0) таблицы, которая не хранится в базе данных только для чтения, необходим уникальный индекс. Можно обойти это требование, заставив Adaptive Server выбрать неуникальный индекс или сканировать саму таблицу, например следующим образом:

```
select * from имя_таблицы (index имя_таблицы)
```

Если во время сканирования с базовой таблицей выполняются какие-либо операции, оно может быть аварийно завершено.

read committed | 1
: По умолчанию уровнем изоляции транзакций в Adaptive Server является read committed, или 1, допускающий разделяемые блокировки чтения.

repeatable read | 2
: предотвращает невоспроизводимые чтения.

serializable | 3

Задает уровень изоляции 3. Сервер Adaptive Server применяет настройку holdlock во всех операциях select и readtext в транзакции. При этом блокировки чтения для запросов удерживаются до завершения транзакции. Если также установлен связанный режим транзакций, этот уровень изоляции также будет использоваться для любого оператора, выполняющего выборку или изменение данных, который неявно начинает транзакцию.

transactional_rpc

Задает способ обработки вызовов удаленных процедур. Если этот параметр установлен в on, то если транзакция отложена, вызов удаленной процедуры координируется самим Adaptive Server. Если этот параметр установлен в off, вызов удаленной процедуры обрабатывается межузловым обработчиком Adaptive Server. Значение по умолчанию наследуется от параметра конфигурации enable xact coordination.

Примеры

Пример 1. Для всех запросов возвращает описание плана обработки, но не выполняет его:

```
set showplan, noexec on
go
select * from publishers
go
```

Пример 2. Задает ограничение в 100 байтов на размер данных типа text и image, возвращаемых командой select:

```
set textsize 100
```

Пример 3. Для всех команд insert, update, delete и select Adaptive Server прекращает обработку запроса после обработки первых четырех строк. Например:

```
select title_id, price from titles
title_id  price
--------  ----------
BU1032         19.99
BU1111         11.95
BU2075          2.99
BU7832         19.99

(4 rows affected)

set rowcount 4
```

Пример 4. Активирует преобразование набора символов (при этом будет использоваться преобразование по умолчанию, основанное на наборе символов, используемом клиентом). Если символы не могут быть преобразованы в набор символов клиента, Adaptive Server уведомит об этом клиента или приложение:

```
set char_convert on with error
```

Пример 5. Пользователь, выполняющий эту команду, теперь работает на сервере под именем “mary” и идентификатором этого пользователя:

```
set proxy "mary"
```

Пример 6. Альтернативный способ записи примера 5:

```
set session authorization "mary"
```

Пример 7. Возвращает пять строк для каждого последующего оператора fetch с курсором *test_cursor*, выполняемого клиентом:

```
set cursor rows 5 for test_cursor
```

Пример 8. Вставляет значение 100 в столбец IDENTITY таблицы stores_south, а затем запрещает дальнейшие явные вставки в этот столбец. Обратите внимание на использование ключевого слова syb_identity; Adaptive Server заменяет это ключевое слово именем столбца IDENTITY:

```
set identity_insert stores_south on
go
insert stores_south (syb_identity)
values (100)
go
set identity_insert stores_south off
go
```

Пример 9. Удерживает блокировки чтения для каждого оператора select в транзакции на протяжении этой транзакции:

```
set transaction isolation level 3
```

Пример 10. Отключает роль системного администратора для пользователя в текущем сеансе:

```
set role "sa_role" off
```

Пример 11. Указывает Adaptive Server выдавать предупреждающие сообщения при использовании расширения Transact-SQL:

```
set fipsflagger on
```

В этом случае при использовании нестандартного SQL, например:

```
use pubs2
go
```

Adaptive Server выдаст сообщение:

```
SQL statement on line number 1 contains Non-ANSI text.
The error is caused due to the use of use database.
```

Пример 12. Указывает Adaptive Server обрабатывать операнды со значением NULL в операциях сравнения (= и !=) и в агрегатных функциях согласно начальному уровню стандарта SQL92:

```
set ansinull on
```

Если set ansinull установлен в on, агрегатные функции и строки с агрегированными значениями инициируют следующее предупреждение SQLSTATE, если в одном или нескольких столбцах или строках находятся значения NULL:

```
Warning - null value eliminated in set function
```

Если в сравнении на равенство или на неравенство операнд равен NULL, результатом сравнения будет UNKNOWN. Например, следующий запрос не возвращает ни одной строки в режиме ansinull:

```
select * from titles where price = null
```

Если же этот параметр установлен в off, этот запрос возвращает строки, в которых столбец price равен NULL.

Пример 13. Указывает серверу Adaptive Server инициировать исключение при усечении строки типа char, unichar или nchar:

```
set string_rtruncation on
```

Если оператор insert или update должен выполнить усечение строки, Adaptive Server выдаст:

```
string data, right truncation
```

Пример 14. Указывает Adaptive Server воспринимать любую символьную строку, заключенную в двойные кавычки, как идентификатор. Если параметр quoted_identifier установлен в on, то имя таблицы “!*&strange_table” и имя столбца “emp’s_name” являются допустимыми именами идентификаторов:

```
set quoted_identifier on
go
create table "!*&strange_table"
     ("emp's_name" char(10),
     age int)
go
set quoted_identifier off
go
```

Пример 15. Указывает, что исходящие RPC-вызовы по умолчанию обрабатываются службами Component Integration Services:

```
set cis_rpc_handling on
```

Пример 16. Указывает, что когда транзакция ожидает обработки, вызовы RPC обрабатываются методами доступа служб Component Integration Services, а не межузловым обработчиком Adaptive Server:

```
set transactional_rpc on
```

Пример 17. Активирует роль doctor_role. Эта команда активирует указанную роль:

```
set role doctor_role on
```

Пример 18. Активирует роль doctor_role, когда пользователь вводит пароль:

```
set role doctor_role with passwd "physician" on
```

Пример 19. Отключает роль doctor_role:

```
set role doctor_role off
```

Пример 20. Задает для параллельных сканирований индексов и несекционированных максимальную степень параллелизма, равную 4:

```
set scan_parallel_degree 4
```

Пример 21. Последующие команды в сеансе или хранимой процедуре будут ждать пять секунд для получения блокировки, а затем завершатся с сообщением об ошибке:

```
set lock wait 5
```

Пример 22. Последующие команды в сеансе или хранимой процедуре выдадут сообщение об ошибке, если не смогут сразу получить необходимые блокировки:

```
set lock nowait
```

Пример 23. Последующие команды в сеансе или хранимой процедуре могут ждать получения блокировок сколь угодно долго:

```
set lock wait
```

Пример 24. Эта команда указывает, что все следующие команды в сеансе запускаются с уровнем изоляции транзакций “repeatable reads”:

```
set transaction isolation level 2
```

Пример 25. Активирует запись абстрактных планов в группу dev_plans:

```
set plan dump dev_plans on
```

Пример 26. Активирует загрузку абстрактных планов из группы dev_plans для запросов в текущем сеансе:

```
set plan load dev_plans on
```

Пример 27. Назначает этому пользователю:

- имя клиента alison;
- имя узла money1;
- имя приложения webserver2.

```
set clientname 'alison'
set clienthostname 'money1'
set clientapplname 'webserver2'
```

Использование

- Некоторые параметры set можно группировать следующим образом:
  - Параметры parseonly, noexec, prefetch, showplan, rowcount и nocount управляют способом выполнения запроса. Не имеет смысла одновременно устанавливать параметры parseonly on и noexec on. Параметр rowcount по умолчанию равен 0 (возвращаются все строки), а другие параметры – off.
  - Параметры statistics выводят статистику производительности после каждого запроса. Значение по умолчанию для параметров statistics – off. Дополнительную информацию о параметрах noexec, prefetch, showplan и statistics см. в книге *Руководство по настройке производительности*.
  - В инструкции set можно указать до 1024 столбцов для обновления, используя для указания новых значений литералы, переменные или выражения, возвращенные подзапросом.
  - Для интерпретации результатов, возвращенных Adaptive Server, в библиотеке DB-Library используются параметры offsets и procid. По умолчанию эти параметры равны on.
  - Параметры datefirst, dateformat и language влияют на функции дат, порядок частей даты и отображение сообщений. Если эти параметры устанавливаются внутри триггера или хранимой процедуры, им не возвращаются предыдущие значения после завершения триггера или процедуры.

    Если используется язык по умолчанию (us_English), то параметр datefirst равен 1 (воскресенье), dateformat – *mdy*, а сообщения отображаются на английском. Для одних языков (включая us_english) по умолчанию воскресенье=1, понедельник=2 и т. д., а для других – понедельник=1, вторник=2 и т. д.

Если была выполнена команда set language, то первый день недели и формат даты будут взяты из языка, указанного в этой команде, если эти параметры не были установлены ранее в текущем сеансе командами set datefirst или set dateformat.

- Параметры cursor rows и close on endtran определяют, как будет происходить обработка курсоров. По умолчанию параметр cursor rows равен 1 для любого курсора, а параметр close on endtran – off.
- Параметры chained и transaction isolation level позволяют Adaptive Server обрабатывать транзакции в соответствии со стандартами SQL.

Параметры fipsflagger, string_rtruncation, ansinull, ansi_permissions, arithabort и arithignore определяют различные аспекты обработки ошибок сервером Adaptive Server, а также совместимости со стандартами SQL.

Примечание. Параметры arithabort и arithignore были переопределены для версии 10.0 и более поздних. Если эти параметры используются в приложениях, следует убедиться, что они по-прежнему обеспечивают желаемый результат.

- Параметры cis_rpc_handling и transactional_rpc можно использовать, только когда службы Component Integration Services включены.
- Если параметр quoted_identifier установлен в on, идентификатор не нужно заключать в двойные кавычки, если в синтаксисе оператора требуется, чтобы строка в кавычках содержала идентификатор. Например:

```
set quoted_identifier on
create table "1one" (c1 int)
```

Однако для функции object_id требуется указать строку, поэтому в ней имя таблицы необходимо заключить в одинарные кавычки:

```
select object_id('1one')
-----------------------
  896003192
```

Чтобы включить в имя идентификатора, заключенное в кавычки, двойную кавычку, нужно указать две двойные кавычки подряд:

```
create table "embedded""quote" (c1 int)
```

Впрочем, нет необходимости указывать две двойные кавычки подряд, если синтаксис оператора требует, чтобы имя объекта было строкой:

```
select object_id('embedded"quote')
```

- Параметры parallel_degree и scan_parallel_degree ограничивают степень параллелизма для запросов, если Adaptive Server сконфигурирован для параллельного выполнения. Эти параметры подсказывают оптимизатору, что нужно ограничить количество параллельных запросов и использовать меньше рабочих процессов, чем разрешено параметрами конфигурации. Установка этих параметров в 0 восстанавливает значения, заданные на уровне сервера.

  Если указанное значение превышает значение соответствующего параметра конфигурации, СУБД Adaptive Server выдает предупреждение и использует значение, заданное параметром конфигурации.

- Большинство параметров, установленных командой set внутри триггера или хранимой процедуры, возвращаются к своим прежним значениям после выполнения триггера или процедуры.

  Следующим параметрам не возвращаются их прежние значения после выполнения процедуры или триггера. Вместо этого заданные значения сохраняются в течение всего сеанса Adaptive Server или до их явного изменения:

  - datefirst
  - dateformat
  - identity_insert
  - language
  - quoted_identifier

- Если в команде set указано несколько параметров и какой-то параметр указан с синтаксической ошибкой, то все параметры, идущие после ошибки, игнорируются. Тем не менее, параметрам, указанным до ошибки, присваиваются новые значения.

- Назначения пользователю имени клиента, узла или приложения, действуют только в текущем сеансе. При следующем соединении пользователя с сервером эти назначения придется сделать заново. Хотя новые имена отображаются в таблице sysprocesses, они не используются для проверки полномочий, а в системной процедуре sp_who по-прежнему отображается соединение клиента с исходным регистрационным именем. Дополнительную информацию о настройке пользовательских процессов см. в книге *Руководство по системному администрированию*.

- Все параметры, устанавливаемые командой set (кроме showplan и char_convert), вступают в силу немедленно. Параметр showplan вступает в силу в следующем пакете. Ниже даны два примера использования параметра set showplan on:

```
set showplan on
select * from publishers
go
```

```
pub_id  pub_name              city       state
------- --------------------  ---------- ---
0736    New Age Books         Boston      MA
0877    Binnet & Hardley      Washington  DC
1389    Algodata Infosystems  Berkeley    CA

(3 rows affected)
```

Но:

```
set showplan on
go
select * from publishers
go
QUERY PLAN FOR STATEMENT 1 (at line 1).
    STEP 1
        The type of query is SELECT

        FROM TABLE
            publishers
        Nested iteration
        Table Scan
        Ascending Scan.
        Positioning at start of table.
```

```
pub_id  pub_name              city        state
------  --------------------  ----------  ----
0736    New Age Books         Boston      MA
0877    Binnet & Hardley      Washington  DC
1389    Algodata Infosystems  Berkeley    CA

(3 rows affected)
```

Роли и команда *set*

- Когда пользователь соединяется с Adaptive Server, все предоставленные ему системные роли активируются автоматически. Однако роли, определенные пользователем, которые были ему предоставлены, не активируются автоматически. Чтобы автоматически активировать определенные пользователем роли, выполните системную процеду-

ру sp_modifylogin. Подробности см. в описании процедуры “sp_modifylogin” на стр. 1138. Для включения и отключения ролей используются команды set role имя_роли on или set role имя_роли off .

Например, пользователь с ролью системного администратора может действовать от имени владельца текущей базы данных (и использовать его идентификатор). Чтобы вернуть настоящий идентификатор пользователя, выполните следующую команду:

```
set role "sa_role" off
```

Если пользователь не зарегистрирован в текущей базе данных и в этой базе данных нет роли пользователя-гостя, он не может устанавливать параметр sa_role off.

- Чтобы активировать пользовательскую роль, для которой назначен пароль, необходимо указать этот пароль. Таким образом, в этом случае нужно использовать следующий пароль:

```
set role "имя_роли" with passwd "пароль" on
```

Распределенные транзакции, службы CIS и команда *set*

- Поведение свойства конфигурации cis rpc handling и команд set transactional_rpc изменилось с появлением ASTC. Если в версиях ранее 12.0 активировать параметр cis rpc handling, то *все* вызовы удаленных процедур направляются через библиотеку Client- Library из состава CIS. Таким образом, в этом случае установка параметра transactional_rpc не имела значения: система всегда вела себя так, как если бы этот параметр был установлен в on. Начиная с версии 12.0, это поведение изменилось. Если параметр cis rpc handling установлен в on, а параметр transactional_rpc – в off, вызовы удаленных процедур в транзакции направляются через межузловой обработчик. Вызовы удаленных процедур (RPC), выполненные вне транзакции, направляются через библиотеку Client-Library из состава служб CIS.

- Если службы СУБД Adaptive Server по управлению распределенными транзакциями включены, то можно помещать вызовы удаленных процедур (RPC) внутри транзакций. Такие вызовы удаленных процедур называют *транзакционными*. Работу транзакционного вызова удаленных процедур можно включить в контекст текущей транзакции. Действия, выполненные этой удаленной процедурой, можно зафиксировать или откатить вместе с действиями, совершенными локальной транзакцией.

  Для использования транзакционных вызовов удаленных процедур нужно включить службы CIS и управление распределенными транзакциями с помощью системной процедуры sp_configure,

а затем выполнить команду set transactional_rpc. Если параметр set transactional_rpc установлен в on и транзакция не завершена, вызов удаленной процедуры координируется сервером Adaptive Server (а не межузловым обработчиком).

По умолчанию параметр transactional_rpc равен off. Команда set cis_rpc_handling переопределяет команду set transactional_rpc. Если установить параметр cis_rpc_handling в on, все внешние вызовы удаленных процедур будут обрабатываться службами CIS.

- Дополнительную информацию об использовании команд set transactional_rpc, set cis_rpc_handling и системной процедуры sp_configure см. в книге *Component Integration Services User's Guide*.

Использование команд set proxy и set session authorization

Примечание. Ни роль "sa_role", ни роль "sso_role" не позволяют выполнить команду set proxy login_name, если на это нет явных полномочий. Чтобы пользователь (в том числе администратор безопасности) мог выполнять команду set proxy login_name, ему должны быть явно предоставлены соответствующие полномочия администратором безопасности.

- Прежде чем можно будет выполнять команду set proxy или set session authorization , администратор безопасности должен предоставить соответствующие полномочия из базы данных master.
- Чтобы восстановить прежнее имя пользователя в базе данных, нужно указать в команде set proxy или set session authorization исходное *регистрационное_имя*.
- Команду set proxy или set session authorization нельзя выполнять в транзакции.
- Adaptive Server допускает только один уровень изменения имен пользователей. Поэтому после того, как имя пользователя было изменено с помощью команды set proxy или set session authorization, нужно вернуться к исходному имени пользователя перед тем, как изменить его снова. Предположим, пользователь имеет регистрационное имя ralph. Чтобы создать таблицу от имени mary, представление от имени joe, а затем вернуться к исходному имени пользователя, нужно выполнить следующие команды:

```
set proxy "mary"
    create table mary_sales
    (stor_id  char(4),
    ord_num   varchar(20),
```

```
    date      datetime)
grant select on mary_sales to public
set proxy "ralph"
set proxy "joe"
    create view joes_view (publisher, city,
        state)
    as select stor_id, ord_num, date
    from mary_sales
set proxy "ralph"
```

Использование команды *lock wait*

- По умолчанию задача Adaptive Server, которая не может немедленно установить блокировку, ожидает снятия блокировок несовместимого типа, а затем продолжает обработку. К таким же результатам приведет выполнение команды set lock wait , в которой не указано значение параметра *количество_секунд*.
- Чтобы установить период ожидания блокировки на уровне сервера, нужно выполнить системную процедуру sp_configure с параметром lock wait period.
- Параметр
- lock wait period, а также время ожидания блокировки, установленное на уровне сеанса командой set lock wait nnn, применимо только к пользовательским таблицам. Эти параметры не влияют на системные таблицы.
- Период ожидания блокировки, определенный на уровне сеанса или в хранимой процедуре командой set lock, переопределяет время ожидания блокировки, установленное на уровне сервера.
- Если в команде set lock wait не указано *количество_секунд*, все последующие команды в текущем сеансе будут ждать сколь угодно долго для получения необходимых блокировок.
- Системная процедура

  sp_sysmon выводит количество случаев, в которых задачи не смогли установить нужную блокировку до истечения периода ожидания.

Уровень изоляции транзакций Repeatable-reads

- При уровне изоляции транзакций repeatable-reads (также называемом вторым уровнем изоляции транзакций), все страницы, читаемые командой, блокируются до завершения транзакции.
- Невоспроизводимым чтением называется ситуация, когда одна транзакция считывает строки из таблицы, а другая меняет те же строки и фиксирует изменения до завершения первой транзакции.

Если первая транзакция повторно считает строки, они будут другими, поэтому первоначальное чтение нельзя воспроизвести. На этом уровне изоляции во время транзакции удерживаются разделяемые блокировки, не позволяющие другим транзакциям обновлять заблокированные строки или строки на заблокированных страницах.

Использование статистики, полученной по результатам моделирования

- Статистику, полученную по результатам моделирования, можно загрузить в базу данных с помощью режима simulate утилиты optdiag. Если выполнить в сеансе команду set statistics simulate on, то запросы будут оптимизироваться по смоделированной, а не по фактической статистике для таблицы.

Глобальные переменные, на которые влияет команда *set*

- В таблице 7-34 перечислены глобальные переменные, содержащие сведения о параметрах сеанса, устанавливаемых командой set.

Таблица 7-34. Глобальные переменные, содержащие параметры сеанса

Глобальная переменная	Описание
@@char_convert	Содержит 0, если преобразование набора символов не включено. Содержит 1, если преобразование набора символов включено.
@@isolation	Содержит текущий уровень изоляции, используемый программой Transact-SQL. *@@isolation* принимает значение активного уровня (0, 1 или 3).
@@options	Содержит шестнадцатеричное представление параметров, установленных командой set для сеанса.
@@parallel_degree	Содержит текущую установку максимальной степени параллелизма.
@@rowcount	Содержит количество строк, обработанных последним запросом. Для команды, которая не возвращает строк (например, if, update или delete), переменная *@@rowcount* равна 0. Для курсора глобальная переменная *@@rowcount* равна суммарному количеству строк, возвращенных клиенту из набора результатов курсора (вплоть до последней команды fetch). Переменная *@@rowcount* обновляется, даже если параметр nocount установлен в on.
@@scan_parallel_degree	Содержит текущую настройку максимальной степени параллелизма для сканирований некластерных индексов.
@@textsize	Максимальное количество байтов для данных типа text или image, возвращаемых командой select. Значение по умолчанию для isql равно 32 КБ, а в общем случае оно зависит от клиентской программы. Его можно изменить для сеанса с помощью команды set textsize.
@@tranchained	Содержит текущий режим транзакций, используемый программой Transact-SQL. Переменная *@@tranchained* равна 0 для несвязанного режима и 1 – для связанного.

Использование параметра *fipsflagger* с языком Java в базе данных

- Если параметр fipsflagger установлен в on, Adaptive Server выдает предупреждение, если используется одно из следующих расширений:
  - утилита installjava;
  - команда remove java;
  - объявления столбцов и переменных, которые ссылаются на Java-классы как на типы данных;
  - операторы, ссылающиеся на членов с помощью выражений Java-SQL.
- Состояние параметра fipsflagger не влияет на арифметические выражения, выполняемые Java-методами.
- Дополнительную информацию об использовании языка Java в базе данных см. в книге *Java in Adaptive Server Enterprise*.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

В ранних версиях Adaptive Server поведение Transact-SQL отличается от предусмотренного стандартом SQL92. Для всех приложений предкомпилятора со встроенным SQL по умолчанию включено поведение, совместимое со стандартом SQL. Чтобы обеспечить соответствие этому стандарту для других приложений, можно использовать команду set для установки параметров, перечисленных в таблице 7-35.

Таблица 7-35. Значения параметров set, обеспечивающие совместимость с базовым уровнем стандарта SQL92

Параметр	Значение
ansi_permissions	on
ansinull	on
arithabort	off
arithabort numeric_truncation	on
arithignore	off
chained	on
close on endtran	on
fipsflagger	on
quoted_identifier	on
string_rtruncation	on
transaction isolation level	3

Полномочия

Как правило, команды set по умолчанию могут выполнять все пользователи – для их использования не нужно особых полномочий. К исключениям относятся команды set role, set proxy и set session authorization.

Пользователь может выполнять команду set role только с ролями, предоставленными ему системным администратором или администратором безопасности. Если пользователь вошел в базу данных только потому, что он имеет определенную роль, он не может отключить эту роль во время работы этой базы данных. Например, если пользователь не прошел обычную авторизацию для использования базы данных info_plan, но работает с ней как системный администратор, то Adaptive Server выдаст сообщение об ошибке при попытке установки параметра sa_role в off в базе данных info_plan.

Чтобы пользователь мог выполнять команду set proxy или set session authorization, администратор безопасности должен предоставить ему соответствующие полномочия.

См. также

Команды create trigger, fetch, insert, grant, lock table, revoke

Функции convert

Утилиты isql, optdiag

setuser

Описание

Позволяет владельцу базы работать под именем другого пользователя.

Синтаксис

setuser ["*имя_пользователя*"]

Примеры

Владелец базы данных временно принимает имя пользователя Mary в базе данных, чтобы предоставить пользователю Joe полномочия на таблицу authors, которой владеет Mary:

```
setuser "mary"
go
grant select on authors to joe
setuser
go
```

Использование

- С помощью команды setuser владелец базы данных может принять имя другого пользователя, чтобы изменить объект базы данных другого пользователя, предоставить полномочия, создать объект или выполнить какое-то другое действие.
- Когда владелец базы данных принял имя другого пользователя с помощью команды setuser, Adaptive Server будет проверять полномочия пользователя, от имени которого действует владелец, а не полномочия владельца базы данных. Пользователь, от имени которого действует владелец, должен быть указан в таблице sysusers базы данных.
- Команда setuser влияет на полномочия только в локальной базе данных. Она не влияет на вызовы удаленных процедур и обращение к объектам в других базах данных.
- Команда setuser остается в силе до выполнения другой команды setuser или до смены текущей базы данных командой use.
- Если в команде setuser не указать имя пользователя, будет восстановлено исходное имя владельца базы данных.
- Команда setuser позволяет системному администратору создавать объекты, которые будут принадлежать другому пользователю. Но поскольку системный администратор работает вне системы полномочий, он не может применить команду setuser для получения полномочий другого пользователя.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду setuser может выполнять только владелец базы данных, причем эти полномочия не подлежат передаче.

См. также

Команды grant, revoke, use

shutdown

Описание

Останавливает сервер Adaptive Server, с которого выполнена команда, его локальный Backup Server или удаленный Backup Server. Эту команду может выполнять только системный администратор.

Синтаксис

shutdown [*имя_сервера*] [with {wait | nowait}]

Параметры

имя_сервера

Логическое имя Backup Server, которое определено в системной таблице sysservers сервера Adaptive Server. Этот параметр необязателен при остановке локального Adaptive Server.

with wait

Действует по умолчанию. Обеспечивает корректную остановку Adaptive Server или Backup Server.

with nowait

Обеспечивает немедленную остановку Adaptive Server или Backup Server без ожидания того, когда завершится выполнение текущих операторов.

Примечание. В результате выполнения команды shutdown with nowait между значениями столбца IDENTITY могут возникнуть разрывы.

Примеры

Пример 1. Остановка сервера Adaptive Server, с которого выполняется команда shutdown:

```
shutdown
```

Пример 2. Немедленная остановка Adaptive Server:

```
shutdown with nowait
```

Пример 3. Остановка локального сервера Backup Server:

```
shutdown SYB_BACKUP
```

Пример 4. Остановка удаленного Backup Server REM_BACKUP:

```
shutdown REM_BACKUP
```

Использование

- Если параметр nowait не используется, команда shutdown пытается корректно остановить Adaptive Server, выполняя следующие действия.
  - Отключение регистрационных имен (за исключением системного администратора).
  - Выполнение контрольных точек в каждой базе данных.
  - Ожидание завершения работы операторов SQL или хранимых процедур.

Остановка сервера без указания параметра nowait сокращает объем работы, который придется выполнять при автоматическом восстановлении.

- Если параметр nowait не указан, команда shutdown backup_server ожидает завершения активных операций загрузки и/или резервного копирования. Если команда shutdown была выполнена с сервером Backup Server, нельзя будет выполнять операции загрузки или резервного копирования, использующие этот сервер.
- Команда shutdown with nowait используется только в крайнем случае. На сервере Adaptive Server перед выполнением команды shutdown with nowait необходимо выполнить команду checkpoint.
- Командой shutdown можно остановить только локальный Adaptive Server; ею нельзя остановить удаленный Adaptive Server.
- Пользователь может остановить сервер Backup Server только при выполнении следующих условий:
  - Сервер указан в таблице sysservers для этого пользователя. Для добавления записей в таблицу sysservers используется процедура sp_addserver.
  - Сервер указан в файле интерфейсов для сервера Adaptive Server, на котором выполняется команда.
- Определить имя, под которым сервер Backup Server известен серверу Adaptive Server, можно с помощью процедуры sp_helpserver. В качестве параметра *имя_сервера* нужно указать имя сервера Backup Server, которое эта процедура возвращает в столбце name (а не network_name). Например:

```
sp_helpserver
name        network_name   status                                id
----------  -------------  ----------------------------------- --
REM_BACKUP  WHALE_BACKUP   timeouts, no net password encryption 3
SYB_BACKUP  SLUG_BACKUP    timeouts, net password encryption    1
eel         eel                                                 0
whale       whale          timeouts, no net password encryption 2
```

Для остановки удаленного сервера Backup Server с именем WHALE_BACKUP должна быть введена следующая команда:

```
shutdown REM_BACKUP
```

Стандарты	Уровень соответствия стандарту SQL92: расширение Transact-SQL.
Полномочия	Команду shutdown по умолчанию могут выполнять системные администраторы. Это полномочие не может быть передано другим пользователям.
См. также	**Команды** alter database
	Системные процедуры sp_addserver, sp_helpserver

truncate table

Описание

Удаляет все строки из таблицы.

Синтаксис

truncate table [[*база_данных*.]*владелец*.]*имя_таблицы*

Параметры

имя_таблицы

Имя очищаемой таблицы. Если таблица расположена в другой базе данных, то должно быть указано имя базы данных. Если в базе данных существуют несколько таблиц с тем же именем, то должно быть задано имя владельца. По умолчанию *владелец* – это текущий пользователь, а *база_данных* – текущая база данных.

Примеры

Удаление всех данных из таблицы authors:

```
truncate table authors
```

Использование

- Команда truncate table удаляет все строки из таблицы. Структура таблицы и все индексы сохраняются, пока не будет выполнена команда drop table. Правила, значения по умолчанию и ограничения, связанные со столбцами, остаются связанными, а триггеры продолжают действовать.
- Сервер Adaptive Server больше не использует страницы данных о распределении; теперь статистическая информация хранится в таблицах sysstatistics и systabstats.

  При выполнении команды truncate table статистическая информация больше не удаляется, так что теперь не нужно выполнять команду update statistics после добавления данных.

  Команда truncate table не удаляет статистическую информацию для таблицы.
- Команда truncate table – более быстрый эквивалент команды delete без инструкции where. Команда delete удаляет строки по одной и каждую операцию удаления строки регистрирует в журнале как транзакцию; команда truncate table освобождает целые страницы данных и делает меньше записей в журнале. Обе команды (и delete, и truncate table) делают пространство, которое было занято удаляемыми данными и связанными с ними индексами, доступным для последующего использования.
- Поскольку операции удаления отдельных строк не записываются в журнал, команда truncate table не может вызывать срабатывание триггера.

- Нельзя выполнять команду truncate table, если другая таблица содержит строки, ссылающиеся на очищаемую таблицу. Сначала необходимо удалить эти строки из внешней таблицы или очистить ее командой truncate, а затем очистить первичную таблицу.
- Нельзя применять команду truncate table к секционированной таблице. Сначала необходимо преобразовать ее в таблицу без секций с помощью инструкции unpartition команды alter table, а затем выполнить команду truncate table.

  Чтобы удалить все строки из секционированной таблицы без предварительной отмены ее секционирования, можно использовать команду delete без инструкции where. Эта команда, как правило, выполняется медленнее команды truncate table, поскольку она удаляет строки по одной и каждую операцию delete регистрирует в журнале.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем.

Полномочия

Команду create existing table по умолчанию может выполнять владелец таблицы. Это полномочие не может быть передано другим пользователям. Чтобы иметь возможность очищать таблицы системного аудита (sysaudits_01, sysaudits_02, sysaudits_03 и т.д. до sysaudits_08), необходимо быть администратором безопасности.

См. также

Команды create trigger, delete, drop table

union

Описание

Возвращает единый набор результатов, объединяющий результаты двух или более запросов. Повторяющиеся строки исключаются из набора результатов, если не указано ключевое слово all.

Синтаксис

```
select список_выборки [into инструкция]
                [from инструкция] [where инструкция]
                [group by инструкция] [having инструкция]
        [union [all]
            select список_выборки
                [from инструкция] [where инструкция]
                [group by инструкция] [having инструкция] ]...
        [order by инструкция]
        [order by инструкция]
```

Параметры

union

Создает объединение результирующих наборов двух операторов select.

all

Включает все строки в результаты, не удаляя повторяющиеся строки.

into

Создает новую таблицу на основе столбцов, указанных в списке выборки, и строк, выбранных инструкцией where. В операторе объединения только первый запрос может содержать инструкцию into.

Примеры

Пример 1. Результирующий набор включает содержимое столбцов stor_id и stor_name таблиц sales и sales_east:

```
select stor_id, stor_name from sales
union
select stor_id, stor_name from sales_east
```

Пример 2. Инструкция into в первом запросе указывает, что в таблицу results должен быть помещен окончательный набор результатов, полученный в результате объединения указанных столбцов таблиц publishers, stores и stores_east:

```
select pub_id, pub_name, city into results
from publishers
union
select stor_id, stor_name, city from stores
union
select stor_id, stor_name, city from stores_east
```

Пример 3. Сначала создается объединение указанных столбцов таблиц sales и sales_east. Затем эти результаты объединяются с данными из таблицы publishers. Наконец, этот результат объединяется с данными из таблицы authors:

```
select au_lname, city, state from authors
union
((select stor_name, city, state from sales
union
select stor_name, city, state from sales_east)
union
select pub_name, city, state from publishers)
```

Использование

- Общее количество таблиц, указанных во всех частях запроса с оператором union, не должно превышать 256.
- Оператор union можно использовать в команде select, например:

```
create view
select * from Jan1998Sales
union all
select * from Feb1998Sales
union all
```

- Инструкции order by и compute разрешены только в конце оператора union для упорядочения конечных результатов или вычисления итоговых значений.
- Инструкции group by и having могут использоваться только внутри индивидуальных запросов; их нельзя применять для группирования или фильтрации конечного набора результатов.
- По умолчанию оператор SQL, содержащий операторы union, обрабатывается слева направо.
- Поскольку union – бинарная операция, выражения с более чем двумя запросами должны заключаться в круглые скобки для указания порядка обработки.
- Первый запрос в операторе union может содержать инструкцию into, которая создает таблицу, в которую будет помещен окончательный набор результатов. Оператор into должен быть в первом запросе. Если он указан в каком-либо другом запросе, то будет выдано сообщение об ошибке (см. пример 2).
- Оператор union можно использовать внутри оператора insert...select, например:

```
insert into sales.overall
  select * from sales
  union
  select * from sales_east
```

- Все списки выборки в операторе SQL должны иметь одинаковое количество выражений (имен столбцов, арифметических выражений, агрегатных функций и т.п.). Например, следующий оператор недопустим, потому что первый список выборки содержит больше выражений, чем второй:

```
/* Пример неправильной команды--списки выборки */ /*
несовместимы */
select au_id, title_id, au_ord from titleauthor
union
select stor_id, date from sales
```

- Соответствующие столбцы в списках выборки команд union должны располагаться в одном и том же порядке, поскольку union сравнивает столбцы по одному в порядке, указанном в индивидуальных запросах.

- Имена столбцов в таблице, полученной в результате выполнения оператора union, берутся из *первого* индивидуального запроса. Если необходимо определить новый заголовок столбца для результирующего набора, то это должно быть сделано в первом запросе. Ссылаться на столбец результирующего набора по новому имени (например в инструкции order by) можно, только если ссылка на это имя была в первом операторе select. Например, следующий запрос корректен:

```
select Cities = city from stores
union
select city from stores_east
order by Cities
```

- Описания столбцов, которые являются частью операции union, необязательно должны быть идентичными. В таблице 7-36 приведены правила относительно типов данных и определений соответствующих друг другу столбцов в индивидуальных запросах.

Таблица 7-36. Результирующие типы данных в операциях union

Типы данных столбцов в индивидуальных запросах	Тип данных соответствующего столбца в таблице, полученной в результате операции union
Несовместимые типы данных (Adaptive Server не преобразует типы данных неявным образом).	Adaptive Server возвращает ошибку.

Типы данных столбцов в индивидуальных запросах	Тип данных соответствующего столбца в таблице, полученной в результате операции union
Оба – символьные типы фиксированной длины (длина первого – L1, а второго – L2).	Символьный тип фиксированной длины, равной большему из двух значений L1 и L2.
Оба – двоичные типы фиксированной длины L1 и L2.	Двоичный тип фиксированной длины, равной большему из двух значений L1 и L2.
Один или оба – символьные типы переменной длины.	Символьный тип переменной длины, равной максимальной из длин столбцов в индивидуальных запросах.
Один или оба – двоичные типы переменной длины.	Двоичный тип переменной длины, равной максимальной из длин столбцов в индивидуальных запросах.
Оба – числовые типы (например smallint, int, float, money).	Тип данных с точностью, равной большей из двух точностей входных столбцов. Например, если столбец в таблице A имеет тип int, а соответствующий столбец в таблице B – тип float, то столбец в результирующей таблице будет иметь тип float, т.к. тип float имеет более высокую точность, чем int.
Оба столбца не допускают значений NULL.	Не допускает значения NULL.

Ограничения

- Оператор union нельзя использовать в подзапросах.
- Оператор union нельзя использовать с инструкцией for browse.
- Оператор union нельзя использовать в запросах, которые выбирают данные типа text или image.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем.

Ниже приведен список расширений TransactSQL:

- использование оператора union в инструкции select команды insert;
- указание новых заголовков столбцов в инструкции order by команды select, содержащей оператор union.

См. также

Команды compute, declare, group by и having, order by, select, where

Функции convert

update

Описание

Изменяет данные существующих строк путем добавления данных либо модификации существующих данных.

Синтаксис

```
update [[база_данных.]владелец.]{имя_таблицы | имя_представления}
        set
[[[база_данных.]владелец.]{имя_таблицы.|имя_представления.}]
            имя_столбца1 =
                {выражение1 | NULL | (оператор_select)} |
            имя_переменной1 =
                {выражение1 | NULL | (оператор_select)}
            [, имя_столбца2 =
                {выражение2 | NULL | (оператор_select)}]... |
            [, имя_переменной2 =
                {выражение2 | NULL | (оператор_select)}]...

        [from [[база_данных.]владелец.]{имя_представления [readpast]|
            имя_таблицы [readpast]
                [(index {имя_индекса | имя_таблицы}
                [ prefetchразмер ][lru|mru])]}
            [,[[база_данных.]владелец.]{имя_представления [readpast]|
            имя_таблицы [readpast]
                [(index {имя_индекса | имя_таблицы}
                [ prefetchразмер ][lru|mru])]}]
        ...]
        [where условия_поиска]
        [plan "абстрактный_план"]

update [[база_данных.]владелец.]{имя_таблицы | имя_представления}
        set
[[[база_данных.]владелец.]{имя_таблицы.|имя_представления.}]
            имя_столбца1 =
                {выражение1 | NULL | (оператор_select)} |
            имя_переменной1 =
                {выражение1 | NULL | (оператор_select)}
            [, имя_столбца2 =
                {выражение2 | NULL | (оператор_select)}]... |
            [, имя_переменной2 =
                {выражение2 | NULL | (оператор_select)}]...
        where current of имя_курсора
```

Параметры

имя_таблицы | *имя_представления*

Имя обновляемой таблицы или представления. Если таблица или представление находятся в другой базе данных, должно быть указано имя базы данных. Если в базе данных существует несколько таблиц или представлений с данным именем, необходимо указать имя владельца. По умолчанию *владелец* – текущий пользователь, а *база_данных* – текущая база данных.

set
Указывает имя столбца или имя переменной и присваивает ей новое значение. Это значение может быть выражением или NULL. Если перечислено несколько имен столбцов или имен и значений переменных, они должны быть разделены запятыми.

from
Использует данные из других таблиц или представлений для изменения строк в обновляемой таблице или представлении.

readpast
Предписывает команде update изменять в таблицах с блокировкой строк только незаблокированные строки, а в таблицах с блокировкой страниц – только строки на незаблокированных страницах. Команда update...readpast пропускает заблокированные строки или страницы, не выдавая сообщений и не дожидаясь снятия блокировок.

where
Стандартная инструкция where (см. раздел where).

index {имя_индекса | имя_таблицы}
Параметр *имя_индекса* указывает индекс, используемый для доступа к таблице *имя_таблицы*. Этот параметр нельзя использовать при обновлении представления.

prefetch размер
Задает размер блока ввода-вывода в килобайтах для таблиц, связанных с кэшами для объемных операций ввода-вывода. Этот параметр нельзя использовать при обновлении представления. Системная процедура sp_helpcache отображает допустимые размеры для кэша, с которым связан объект, или для кэша по умолчанию. Для конфигурирования размера кэша данных используется системная процедура sp_cacheconfigure.

При использовании параметра prefetch и указании размера предварительной выборки (*размер*) необходимо учитывать, что ее минимальный размер равен размеру логической страницы. Допустимые значения размера предварительной выборки можно получить путем умножения размера логической страницы на степени двойки от 1 до 3. Эти допустимые значения параметра prefetch приведены в следующей таблице (в килобайтах).

Размер логической страницы	Размеры предварительной выборки
2	2, 4, 8 16
4	4, 8, 16, 32

Размер логической страницы	Размеры предварительной выборки
8	8, 16, 32, 64
16	16, 32, 64, 128

Указанное в запросе значение размера предварительной выборки носит лишь рекомендательный характер. Чтобы иметь возможность задавать размер, необходимо сконфигурировать кэш данных с указанием этого размера. Если этого не сделать, будет использоваться значение параметра prefetch, заданное по умолчанию.

Использование параметра prefetch для удаленных серверов невозможно, если включены службы Component Integration Services.

lru | mru

Задает стратегию замены буферов для таблицы. Параметр lru предписывает оптимизатору считывать таблицу в кэш по цепочке MRU/LRU (most recently used/least recently used – недавно/давно использованный). Параметр mru используется для удаления буфера из кэша и его замены следующим буфером для таблицы. Этот параметр при обновлении представления использовать нельзя.

where current of

Обновляет строку таблицы или представления, находящуюся в текущей позиции курсора *имя_курсора*.

имя_индекса

Имя обновляемого индекса. Если имя индекса не задано, в указанной таблице обновляется статистика распределения для всех индексов.

plan "абстрактный_план"

Определяет абстрактный план для оптимизации запроса. Это может быть полный или частичный план, заданный на языке абстрактных планов. Дополнительную информацию см. в главе 30, “Руководство по написанию абстрактных планов запросов”, книги *Руководство по настройке производительности*.

Примеры

Пример 1. Все авторы с фамилией McBadden в таблице authors теперь имеют фамилию MacBadden:

```
update authors
set au_lname = "MacBadden"
where au_lname = "McBadden"
```

Пример 2. Изменение столбца total_sales, чтобы он отражал недавние продажи, записанные в таблицы sales и salesdetail. Предполагается, что только один набор продаж записан для определенной книги на определенную дату и обновления являются текущими:

```
update titles
set total_sales = total_sales + qty
from titles, salesdetail, sales
where titles.title_id = salesdetail.title_id
    and salesdetail.stor_id = sales.stor_id
    and salesdetail.ord_num = sales.ord_num
    and sales.date in
        (select max(sales.date) from sales)
```

Пример 3. Изменение цены книги в таблице titles, на которую в данный момент указывает курсор title_crsr, до $24,95:

```
update titles
set price = 24.95
where current of title_crsr
```

Пример 4. Нахождение строки, для которой поле IDENTITY равно 4, и изменение цены книги до $18,95. Сервер Adaptive Server заменяет ключевое слово syb_identity именем столбца IDENTITY:

```
update titles
set price = 18.95
where syb_identity = 4
```

Пример 5. Обновление таблицы titles с использованием объявленной переменной:

```
declare @x money
select @x = 0
update titles
    set total_sales = total_sales + 1,
    @x = price
    where title_id = "BU1032"
```

Пример 6. Обновление строк, которые не заблокированы другими задачами:

```
update salesdetail set discount = 40
        from salesdetail readpast
    where title_id like "BU1032"
        and qty > 100
```

Использование

- Команда update используется для изменения значений уже вставленных строк. Команда insert используется для добавления новых строк.

- В операторе update можно указывать до 15 таблиц.
- Оператор update взаимодействует с параметрами ignore_dup_key, ignore_dup_row и allow_dup_row, задаваемыми командой create index. Дополнительную информацию см. в описании команды create index.
- Можно определять триггер, который будет выполнять указанное действие при выполнении команды update для заданной таблицы или для заданного столбца таблицы.

Использование переменных в операторах *update*

- В инструкции set оператора update можно присваивать значения переменным (так же, как в операторе select).
- Перед использованием переменной в операторе update необходимо объявить переменную командой declare и инициализировать ее командой select, как показано в примере 5.
- Присваивание значений переменным происходит для каждой строки, обновляемой оператором update.
- Если переменная указана справа от присваивания в операторе update, текущее значение этой переменной изменяется после обновления каждой строки. **Текущее значение** – это значение переменной перед обновлением текущей строки. В следующем примере показано, как текущее значение изменяется по мере обновления каждой строки.

  Допустим, выполняются следующие операторы:

```
declare @x int
select @x=0
update table1
    set C1=C1+@x, @x=@x+1
    where column2=xyz
```

  Значение C1 перед обновлением равно 1. В следующей таблице показано, как текущее значение переменной *@x* изменяется после каждого обновления:

Строка	Начальное значение C1	Начальное значение переменной @x	Вычисления: C1+@x= обновленное C1	Новое значение C1	Вычисления: @x+1= новое @x	Новое значение @x
A	1	0	1+0	1	0+1	1
B	1	1	1+1	2	1+1	2
C	2	2	2+2	4	2+1	3
D	4	3	4+3	7	3+1	4

- Если в одном операторе update происходит присваивание значений нескольким переменным, значения, присваиваемые переменным, могут зависеть от их порядка в списке присваивания, но это не всегда так. Поэтому не следует полагаться на порядок записи операций при определении присвоенных значений.
- Если возвращается несколько строк и переменной присваивается значение, не являющееся агрегатным по столбцу, конечное значение этой переменной будет равно значению последней обновленной строки в этом столбце, и поэтому оно вряд ли будет полезным.
- Оператор update, присваивающий значения переменным, не обязан задавать новые значения для обновляемых строк.
- Если в таблице нет подходящих строк для обновления, значение переменной не присваивается.
- На переменную, значение которой присваивается в операторе update, нельзя ссылаться в подзапросе того же оператора update (независимо от того, в каком месте оператора update находится данный подзапрос).
- На переменную, значение которой присваивается в операторе update, нельзя ссылаться в инструкциях where или having того же оператора update.
- Если новые значения выбираются на основе соединения таблиц, то переменная, которой присваивается значение в операторе update, использует столбцы из таблицы, которая не обновляется. Конечное значение зависит от порядка соединения, выбранного для обновления, и количества строк из таблицы, с которой соединяется обновляемая таблица, удовлетворяющих условию соединения.
- На обновление переменной не влияет откат оператора update, так как значение обновленной переменной не хранится на диске.

Использование команды *update* с транзакциями

- Если включен режим связанных транзакций (chained transaction mode on) и в настоящий момент нет активных транзакций, то при выполнении оператора update неявно начинается транзакция. Для завершения обновления необходимо зафиксировать транзакцию (оператором commit) или выполнить откат внесенных изменений (оператором rollback). Например:

```
update stores set city = 'Concord'
    where stor_id = '7066'
if exists (select t1.city, t2.city
    from stores t1, stores t2
```

```
        where t1.city = t2.city
        and t1.state = t2.state
        and t1.stor_id < t2.stor_id)
            rollback transaction
    else
            commit transaction
```

Этот пакет начинает транзакцию (в режиме связанных транзакций) и обновляет строку в таблице stores. Если пакет обновляет строку, содержащую те же значения столбцов city и state, что и другая строка в таблице, производится откат изменений таблицы stores и завершение транзакции. В противном случае изменения фиксируются и транзакция также завершается.

- Adaptive Server разрешает операторам update обновлять одну и ту же строку несколько раз в одной и той же транзакции. Например, оба следующих обновления изменяют цену книги с идентификатором title_id MC2022, так как значение столбца type для нее равно “mod_cook”:

```
begin transaction
update titles
set price = price + $10
where title_id = "MC2222"
update titles
set price = price * 1.1
where type = "mod_cook"
```

Использование соединений в операциях обновления

- Выполнение соединений в инструкции from команды update – расширение синтаксиса SQL (стандарт ANSI) для операций обновления, появившееся в Transact-SQL. Из-за способа обработки оператора update обновления, выполненные одним оператором, не накапливаются. Иными словами, если в операторе update указано соединение и другая таблица в этом соединении имеет несколько совпадающих значений в столбце, по которому происходит соединение, то второе обновление будет основано не на новых значениях от первого обновления, а на исходных значениях. Эти результаты не детерминированы, так как зависят от очередности обработки. Рассмотрим такое соединение:

```
update titles set total_sales = total_sales + qty
    from titles t, salesdetail sd
    where t.title_id = sd.title_id
```

Значение total_sales обновляется только один раз для каждого значения title_id в таблице titles. При этом выбирается значение какой-либо *одной* строки таблицы salesdetail, содержащей данное значение

title_id. Результаты могут зависеть от порядка соединения, используемого в запросе, секционирования таблицы и доступных индексов. Но каждый раз к значению total_sales будет добавлено только одно значение из таблицы salesdetail.

Если требуется прибавить к текущему значению сумму значений во всех строках, для которых значение столбца соединения равно значению текущей строки, нужно использовать подзапрос:

```
update titles set total_sales = total_sales +
    (select isnull(sum(qty),0)
        from salesdetail sd
        where t.title_id = sd.title_id)
    from titles t
```

Использование команды *update* с символьными данными

- Если в результате обновления попытаться занести в символьный столбец переменной длины или в столбец типа text пустую строку (“”), то в такой столбец будет вставлен один пробел. Символьные столбцы фиксированной длины дополняются пробелами до заданной длины.
- Если символьная строка, заносящаяся в столбец переменной длины, состоит не только из пробелов, все завершающие пробелы удаляются. Если строка состоит только из пробелов, она усекается до одного пробела. Строки, длина которых превышает длину, указанную для столбцов типа char, nchar, unichar, varchar, univarchar или nvarchar, усекаются без предупреждения, если только параметр string_rtruncation не установлен в on.
- Команда update, обновляющая столбец типа text, инициализирует этот столбец, присваивает ему текстовый указатель и выделяет как минимум одну текстовую страницу.

Использование команды *update* с курсорами

- Для обновления строки с использованием курсора нужно определить курсор командой declare cursor, а затем открыть его. Именем курсора не может быть параметр Transact-SQL или локальная переменная. Курсор должен быть обновляемым, иначе Adaptive Server возвратит ошибку. Любое обновление набора результатов курсора затрагивает строку базовой таблицы, из которой извлечена строка курсора.
- *Имя_таблицы* или *имя_представления*, заданное в команде update...where current of, должно быть указано в первой инструкции from оператора select, который определяет курсор. Если в этой инструкции from указано более одной таблицы или представления (то есть она содержит соединение), можно указывать только обновляемую таблицу или представление.

После обновления позиция курсора не меняется. При следующем обновлении через курсор будет изменена та же строка, если другой оператор SQL не изменил позицию данного курсора.

- Adaptive Server позволяет обновлять столбцы, которые не указаны в списке столбцов *оператора_select* для курсора, но которые являются частью таблиц, указанных в этом *операторе_select*. Однако когда в инструкции for update объявления курсора указан *список_имен_столбцов*, можно обновлять только столбцы, содержащиеся в этом списке.

Обновление столбцов IDENTITY

- Значения столбца со свойством IDENTITY нельзя изменять ни при обновлении самой таблицы, ни при обновлении через представление, основанное на этой таблице. Чтобы выяснить, был ли столбец определен как IDENTITY, используется процедура sp_help с указанием таблицы, которой принадлежит данный столбец.
- Столбец IDENTITY, выбранный в результирующую таблицу, подчиняется следующим правилам наследования свойства IDENTITY:
  - Если столбец IDENTITY выбирается более одного раза, он определяется в новой таблице как NOT NULL (не допускающий неопределенных значений). Он не наследует свойство IDENTITY.
  - Если столбец IDENTITY выбирается как часть выражения, результирующий столбец, в который будут занесены значения этого выражения, не наследует свойство IDENTITY. Он определяется как NULL (допускающий неопределенные значения), если какой-либо столбец в выражении допускает значения NULL; в противном случае этот столбец определяется как NOT NULL.
  - Если оператор select содержит инструкцию group by или агрегатную функцию, результирующий столбец не наследует свойство IDENTITY. Столбцы, содержащие агрегатную функцию по столбцу IDENTITY, создаются со свойством NULL; остальные столбцы создаются со свойством NOT NULL.
  - Столбец IDENTITY, который выбирается в таблицу оператором, содержащим объединение (оператор union) или соединение (оператор join), не сохраняет свойство IDENTITY. Если оператор union объединяет столбец IDENTITY и столбец, допускающий значения NULL, столбец в новой таблице также будет допускать значения NULL. В противном случае он не будет допускать значений NULL.

Обновление данных через представления

- Нельзя обновлять командой update представления, в определении которых была указана инструкция distinct.
- Если представление создано с использованием инструкции with check option, каждая строка, которая обновляется через это представление, должна оставаться видимой через это представление. Например, представление stores_cal содержит все строки таблицы stores, у которых столбец state имеет значение “CA”. Инструкция with check option проверяет соответствие каждого оператора update условиям выборки данного представления.

  ```
  create view stores_cal
  as select * from stores
  where state = "CA"
  with check option
  ```

  Оператор update вызывает ошибку (как в приведенном ниже примере), если он изменяет значение столбца state на любое значение, отличное от “CA”:

  ```
  update stores_cal
  set state = "WA"
  where store_id = "7066"
  ```

- Если при создании представления указана инструкция with check option, все представления, созданные на его основе, должны удовлетворять условиям выборки базового представления. Каждая строка, обновленная через *производное* представление, должна оставаться видимой через базовое представление.

  Рассмотрим представление stores_cal30, основанное на представлении stores_cal. Это новое представление содержит информацию о магазинах в Калифорнии с условиями платежа “Net 30”.

  ```
  create view stores_cal30
  as select * from stores_cal
  where payterms = "Net 30"
  ```

  Поскольку представление stores_cal было создано с инструкцией with check option, все строки, обновляемые через представление stores_cal30, должны оставаться видимыми через представление stores_cal. Изменения строк, в результате которых столбец state перестает быть равным “CA”, не принимаются.

  Отметим, что представление stores_cal30 было создано без использования инструкции with check option. Поэтому через представление *stores_cal30* можно обновлять строки, в которых значение столбца payterms не равно “Net 30”. Например, следующий оператор update

завершится успешно, хотя строка больше не будет удовлетворять условиям отбора представления stores_cal30.

```
update stores_cal30
set payterms = "Net 60"
where stor_id = "7067"
```

- Нельзя обновлять строку через представление, которое соединяет столбцы двух или более таблиц, если не выполняются сразу оба следующих условия:
  - представление было создано без использования инструкции with check option;
  - все обновляемые столбцы принадлежат одной и той же базовой таблице.
- Операторы update разрешено выполнять для представлений соединения, созданных с инструкцией with check option. Операция обновления завершается ошибкой, если какой-либо из обновляемых столбцов указан в инструкции where в выражении, которое содержит столбцы более чем из одной таблицы.
- При обновлении строки через представление соединения все обновляемые столбцы должны принадлежать одной и той же базовой таблице.

Использование параметров *index*, *prefetch* и *lru | mru*

- Параметры index, prefetch и lru | mru переопределяют значения, выбранные оптимизатором Adaptive Server. Эти параметры следует использовать с осторожностью и всегда проверять их влияние на производительность при помощи команды set statistics io on. Дополнительную информацию об использовании этих параметров см. в книге *Руководство по настройке производительности*.

Использование параметра *readpast*

- Параметр readpast применяется лишь к таблицам с блокировкой только данных. readpast игнорируется, если он указан для таблиц с блокировкой всех страниц.
- Параметр readpast несовместим с параметром holdlock. Если оба этих параметра указаны в одной команде select, выводится сообщение об ошибке и команда аварийно завершается.
- Если в сеансе уровень изоляции равен 3, параметр readpast игнорируется без уведомления.
- Если в сеансе уровень изоляции транзакций равен 0, команды update с параметром readpast не выдают предупреждающих сообщений. В таблицах с блокировкой страниц данных эти команды изменяют все

строки на всех страницах, которые не заблокированы несовместимыми блокировками. В таблицах с блокировкой строк данных они затрагивают все строки, не заблокированные несовместимыми блокировками.

- Если команда update с параметром readpast применяется к двум или более текстовым столбцам и первый проверенный столбец удерживается несовместимой блокировкой, строка пропускается. Если столбец не заблокирован несовместимой блокировкой, команда применяет блокировку и изменяет столбец. Если любое последующее текстовое поле этой строки удерживается несовместимой блокировкой, команда останавливается до тех пор, пока не сможет применить свою блокировку и изменить столбец.
- Дополнительную информацию о блокировках и параметре readpast см. в книге *Руководство по настройке производительности*.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем.

Ниже перечислены расширения TransactSQL:

- Использование инструкции from и полного имени таблицы или столбца – расширения Transact-SQL, которые могут быть обнаружены флагом FIPS. Обновления через представление соединения и представление, чей целевой список содержит выражение, – расширения Transact-SQL, которые не выявляются до этапа выполнения и не маркируются флагом FIPS.
- Использование переменных.
- Инструкция readpast

Полномочия

Полномочия на выполнение команды update по умолчанию принадлежат владельцу процедуры, который может передать их другим пользователям.

Если set ansi_permissions имеет значение on, необходимо обладать полномочием update на обновляемую таблицу, а также полномочием select на все столбцы, перечисленные в инструкции where, и все столбцы в инструкции set. По умолчанию для параметра ansi_permissions задано значение off.

См. также

Команды – alter table, create default, create index, create rule, create trigger, insert, where

Функции – ptn_data_pgs

Системные процедуры – sp_bindefault, sp_bindrule, sp_help, sp_helpartition, sp_helpindex, sp_unbindefault, sp_unbindrule

update all statistics

Описание

Обновляет всю статистическую информацию для указанной таблицы.

Синтаксис

update all statistics *имя_таблицы*

Параметры

имя_таблицы

Имя таблицы, для которой необходимо обновить статистику.

Примеры

Обновление статистики индексов и секционирования для таблицы salesdetail:

```
update all statistics salesdetail
```

Использование

- Команда update all statistics обновляет всю статистическую информацию для указанной таблицы. Adaptive Server хранит статистику о распределении страниц в таблице и использует эти статистические данные при принятии решения, стоит ли применять параллельный просмотр при обработке запросов к секционированным таблицам, и какие индексы использовать при обработке запросов. Оптимизация запросов зависит от точности хранимой статистики.
- Команда update all statistics обновляет статистику для всех столбцов в таблице и статистику секционирования, если таблица секционирована.
- Если таблица не секционирована, команда update all statistics запускает только команду update statistics для таблицы.
- Если таблица секционирована и не имеет индексов, команда update all statistics запускает команду update partition statistics для таблицы. Если таблица секционирована и имеет индексы, команда update all statistics запускает команды update statistics и update partition statistics для таблицы.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду update all statistics по умолчанию может выполнять владелец таблицы. Это полномочие не может быть передано другим пользователям.

См. также

Команды – update statistics, update partition statistics

update partition statistics

Описание

Обновляет информацию о количестве страниц в каждой секции секционированной таблицы.

Синтаксис

update partition statistics *имя_таблицы* [*номер_секции*]

Параметры

имя_таблицы
Имя секционированной таблицы.

номер_секции
Номер секции, информация которой обновляется. Если номер секции не указан, команда update partition statistics обновляет страницы данных во всех секциях указанной таблицы.

Использование

- Сервер Adaptive Server хранит статистику о распределении страниц внутри секционированной таблицы и использует эти статистические данные при принятии решения, стоит ли применять параллельный просмотр при обработке запросов. Оптимизация запросов зависит от точности хранимой статистики. При сбое Adaptive Server информация о распределении может стать неточной.

  Для проверки точности информации о распределении страниц используется функция data_pgs, которая вычисляет количество страниц в таблице:

  ```
  select data_pgs(sysindexes.id, doampg)
     from sysindexes
     where sysindexes.id = object_id("имя_таблицы")
  ```

  Затем необходимо выполнить для таблицы процедуру sp_helpartition и сложить числа в столбце ptn_data_ pgs выходных данных этой процедуры. Сумма всех страниц, сообщенная процедурой sp_helpartition, должна быть немного больше, чем число, возвращенное функцией data_pgs, поскольку sp_helpartition считает также и страницы OAM.

  Если информация о распределении страниц неточна, необходимо выполнить команду update partition statistics для таблицы. Во время обновления информации о распределении команда update partition statistics блокирует страницу OAM и служебную страницу секции.

- При выполнении команды update partition statistics для таблицы, которая содержит данные, или при создании индекса для таблицы с данными столбец controlpage таблицы syspartitions изменяется так, что он указывает на служебную страницу секции.

- Команда update partition statistics обновляет значения служебных страниц, используемые для оценки количества страниц в таблице. Эта статистика используется процедурой sp_helpartition.

Стандарты	Уровень соответствия стандарту SQL92: расширение Transact-SQL.
Полномочия	Команду update partition statistics по умолчанию может выполнять владелец таблицы. Это полномочие не может быть передано другим пользователям.
См. также	*Команды* – alter table, update all statistics *Функции* – ptn_data_pgs *Системные процедуры* – sp_helpartition

update statistics

Описание

Обновляет информацию о распределении ключевых значений в указанных индексах или в указанных столбцах, для всех столбцов в индексе или для всех столбцов в таблице.

Синтаксис

update statistics *имя_таблицы*
[[*имя_индекса*] | [(*список_столбцов*)]]
[using *шаг* values]
[with consumers = *потребители*]

update index statistics *имя_таблицы* [*имя_индекса*]
[using *шаг* values]
[with consumers = *потребители*]

Параметры

имя_таблицы
При использовании в команде update statistics переменная *имя_таблицы* – это имя таблицы, с которой связан индекс. *имя_таблицы* необходимо, поскольку Transact-SQL не требует, чтобы имена индексов в базе данных были уникальны.

имя_индекса
Имя обновляемого индекса. Если имя индекса не задано, в указанной таблице обновляется статистика распределения для всех индексов.

список_столбцов
Список столбцов, разделенных запятыми.

using шаг values
Указывает количество шагов гистограммы. Для столбцов без статистики значение по умолчанию – 20. Если статистика столбца уже существует в таблице sysstatistics, значение по умолчанию – текущее количество шагов.

[with consumers = потребители

]
Указывает количество процессов-потребителей, используемых для сортировки, если указан *список_столбцов* и включена параллельная обработка запросов.

index
Указывает, что статистика обновляется для всех столбцов в индексе.

Примеры

Пример 1. Формирование статистики для столбца price таблицы titles:

```
update statistics titles (price) using 40 values
```

Пример 2. Формирование статистики для всех столбцов во всех индексах таблицы authors:

```
update index statistics authors
```

Пример 3. Формирование статистики для всех столбцов в индексе au_names_ix таблицы authors:

```
update index statistics authors au_names_ix
```

Использование

- Сервер Adaptive Server хранит статистику о распределении ключевых значений в каждом индексе и использует эти статистические данные при выборе индексов для обработки запроса.
- При создании некластерного индекса для таблицы, содержащей данные, команда update statistics автоматически запускается для нового индекса. При создании кластерного индекса для таблицы, содержащей данные, команда update statistics автоматически запускается для всех индексов.
- Оптимизация запросов зависит от точности статистики. Если произошли значительные изменения ключевых значений индекса, то следует перезапустить команду update statistics для этого индекса или столбца. Команда update statistics используется, если значительная часть данных в индексированном столбце была изменена, удалена или добавлена в этот столбец (то есть если есть основания полагать, что распределение ключевых значений изменилось).
- Команда update statistics с именем таблицы и именем индекса обновляет статистику для начального столбца индекса. Если команда update statistics используется только с именем таблицы, она обновляет статистику для начальных столбцов всех индексов таблицы.
- Команда update index statistics с именем таблицы и именем индекса обновляет статистику для всех столбцов в указанном индексе. Если команда update index statistics используется только с именем таблицы, она обновляет статистику для всех столбцов во всех индексах таблицы.
- Указание имени неиндексированного или не первого столбца индекса формирует статистику для этого столбца без создания индекса.
- Указание более одного столбца в списке столбцов формирует или обновляет гистограмму для первого столбца и статистику плотности для всех префиксных подмножеств списка столбцов.
- При использовании команды update statistics для формирования статистики по столбцу или списку столбцов эта команда должна просмотреть таблицу и выполнить сортировку.
- Инструкция with consumers предназначена для применения к секционированным таблицам, хранимым на RAID-устройствах, которые воспринимаются сервером Adaptive Server как отдельные устройства ввода-вывода, однако могут обеспечивать высокую пропускную

способность, необходимую для параллельной сортировки. Дополнительную информацию см. в главе 24, “Параллельная обработка запросов”, книги *Руководство по настройке производительности*.

- В таблице 7-37 перечислены типы просмотров, выполняемых во время выполнения команды update statistics, типы применяемых блокировок и ситуации, когда необходимы сортировки.

Таблица 7-37. Блокировки, просмотры и сортировки в ходе обновления статистики

Команда update statistics с указанием	Выполняемые просмотры и сортировки	Блокировки
Имя таблицы		
Таблица с блокировкой всех страниц	Просмотр таблицы, плюс просмотр на уровне листьев каждого некластерного индекса	Уровень 1; разделяемая намеренная блокировка таблицы, разделяемая блокировка текущей страницы
Таблица с блокировкой только данных	Просмотр таблицы плюс просмотр на уровне листьев каждого некластерного индекса и кластерного индекса, если таковой имеется.	Уровень 0; грязное чтение
Имя таблицы и имя кластерного индекса		
Таблица с блокировкой всех страниц	Просмотр таблицы	Уровень 1; разделяемая намеренная блокировка таблицы, разделяемая блокировка текущей страницы
Таблица с блокировкой только данных	Просмотр индекса на листовом уровне	Уровень 0; “грязное” чтение
Имя таблицы и имя некластерного индекса		
Таблица с блокировкой всех страниц	Просмотр индекса на листовом уровне	Уровень 1; разделяемая намеренная блокировка таблицы, разделяемая блокировка текущей страницы
Таблица с блокировкой только данных	Просмотр индекса на листовом уровне	Уровень 0; “грязное” чтение
Имя таблицы и имя столбца		
Таблица с блокировкой всех страниц	Просмотр таблицы; создание рабочей таблицы и ее сортировка	Уровень 1; разделяемая намеренная блокировка таблицы, разделяемая блокировка текущей страницы
Таблица с блокировкой только данных	Просмотр таблицы; создание рабочей таблицы и ее сортировка	Уровень 0; “грязное” чтение

- Команда update index statistics формирует последовательность операций обновления статистики, использующих те же блокировки, просмотры и сортировки, что и эквивалентные команды на уровне индекса и столбца. Например, если у таблицы salesdetail есть некластерный индекс sales_det_ix по столбцу salesdetail(stor_id, ord_num, title_id), команда

  ```
  update index statistics salesdetail
  ```

  выполняет следующие операции update statistics:

  ```
  update statistics salesdetail sales_det_ix
  update statistics salesdetail (ord_num)
  update statistics salesdetail (title_id)
  ```

- Команда update all statistics формирует последовательность операций update statistics для каждого индекса таблицы, за которой идет последовательность операций update statistics для всех неиндексированных столбцов, а затем операция update partition statistics.

- Команда update statistics не выполняется для системных таблиц базы данных master во время перехода на более новую версию. Индексы существуют по столбцам, запросы к которым выполняют большинство системных процедур, потому выполнение команды update statistics для этих таблиц обычно не требуется. Однако выполнение команды update statistics разрешено в отношении всех системных таблиц во всех базах данных, за исключением следующих нестандартных таблиц: syscurconfigs, sysengines, sysgams, syslisteners, syslocks, syslogs, syslogshold, sysmonitors, sysprocesses, syssecmechs, systestlog и systransactions. При выполнении запросов к этим таблицам они строятся из внутренних структур.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду update statistics по умолчанию может выполнять владелец таблицы. Это полномочие не может быть передано другим пользователям. Эту команду может также выполнять владелец базы данных, который может выдать себя за владельца таблицы, выполнив команду setuser.

См. также

Команды – delete statistics

use

Описание

Указывает базу данных, с которой будет работать пользователь.

Синтаксис

use *имя_базы_данных*

Параметры

имя_базы_данных
Имя открываемой базы данных.

Примеры

```
use pubs2
go
```

Теперь текущей является база данных pubs2.

Использование

- Команда use должна быть выполнена до любых обращений к объектам базы данных.
- Команду use нельзя включать в хранимые процедуры или триггеры.
- Процедура

  sp_addalias добавляет псевдоним, который позволяет пользователю работать с базой данных под другим именем для получения доступа к этой базе данных.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Если в базе данных есть учетная запись “guest”, с этой базой могут работать все пользователи. Если база данных не содержит учетной записи “guest”, для работы с этой базой данных необходимо быть ее действительным пользователем, иметь в ней псевдоним, быть системным администратором или администратором безопасности.

См. также

Команды – create database, drop database

Системные процедуры – sp_addalias, sp_adduser, sp_modifylogin

waitfor

Описание

Указывает определенное время, временной интервал или событие для выполнения блока операторов, хранимой процедуры или транзакции.

Синтаксис

waitfor { delay *время* | time *время* | errorexit
| processexit | mirrorexit }

Параметры

delay
Дает указание серверу Adaptive Server ждать, пока не пройдет указанное время (до 24 часов).

time
Дает указание Adaptive Server ждать до указанного времени.

время
Время в одном из приемлемых форматов для данных типа datetime или переменная символьного типа. Нельзя указывать только дату: значение типа datetime должно включать как дату, так и время.

errorexit
Дает указание Adaptive Server ожидать ненормального завершения процесса системного ядра или процесса пользователя.

processexit
Дает указание Adaptive Server ждать, пока процесс ядра или процесс пользователя не завершится по любой причине.

mirrorexit
Дает указание Adaptive Server ждать сбоя зеркальной копии.

Примеры

Пример 1. В 14:20 таблица chess обновляется следующим ходом, и процедура sendmail вставляет строку в таблицу, владельцем которой является пользователь Judy, и сообщает Judy о новом ходе, записанном в таблицу chess:

```
begin
    waitfor time "14:20"
    insert chess(next_move)
        values('Q-KR5')
    execute sendmail 'judy'
end
```

Пример 2. Указание серверу Adaptive Server выводить заданное сообщение через 10 секунд:

```
declare @var char(8)
select @var = "00:00:10"
begin
    waitfor delay @var
```

```
    print "Ten seconds have passed.  Your time
        is up."
end
```

Пример 3. Указание серверу Adaptive Server выводить заданное сообщение после ненормального завершения какого-либо процесса:

```
begin
    waitfor errorexit
    print "Process exited abnormally!"
end
```

Использование

- После выполнения команды waitfor нельзя использовать соединение с сервером Adaptive Server, пока не пройдет заданное время или не произойдет указанное событие.
- Команду waitfor errorexit можно использовать вместе с процедурой, которая удаляет ненормально завершившийся процесс, для освобождения системных ресурсов, занимаемых этим процессом.
- Чтобы узнать, какие процессы были завершены, нужно проверить таблицу sysprocesses процедурой sp_who.
- Время, указываемое в команде waitfor time или waitfor delay, может содержать часы, минуты и секунды. При этом используется формат “чч:ми:сс”, как описано в разделе “Типы данных для представления даты и времени.”

  В следующем примере серверу Adaptive Server дается указание ждать до 16:23:

  ```
  waitfor time "16:23"
  ```

  Следующий оператор дает указание Adaptive Server ждать 1 час 30 минут:

  ```
  waitfor delay "01:30"
  ```

- Изменения в системном времени (например переход на летнее время) могут задержать выполнение команды waitfor.
- Команду waitfor mirrorexit можно использовать внутри программы библиотеки DB-Library для извещения пользователей о сбое зеркальной копии.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду waitfor по умолчанию могут выполнять всем пользователям. Никаких полномочий для ее выполнения не требуется.

См. также

Команды – begin...end

Типы данных – Типы данных для представления даты и времени

Системные процедуры – sp_who

where

Описание

Задает условия поиска в операторах select, insert, update и delete.

Синтаксис

Условия поиска указываются сразу после ключевого слова where в операторах select, insert, update и delete. Если в одном операторе указано несколько условий, то они должны быть соединены с помощью слов and или or.

where [not] *выражение оператор_сравнения выражение*

where [not] *выражение* [not] like "*строка_соответствия*"
[escape "*экранирующий_символ* "]

where [not] *выражение* is [not] null

where [not]
expression [not] between *выражение* and *выражение*

where [not]
выражение [not] in ({*список_значений* | *подзапрос*})

where [not] exists (*подзапрос*)

where [not]
выражение оператор_сравнения
{any | all} (*подзапрос*)

where [not] *имя_столбца оператор_соединения имя_столбца*

where [not] *логическое_выражение*

where [not] *выражение* {and | or} [not] *выражение*

Параметры

not

Служит отрицанием для любого логического выражения или ключевого слова, например like, null, between, in или exists.

выражение

Имя столбца, константа, функция, подзапрос или любая комбинация имен столбцов, констант и функций, связанных арифметическими или поразрядными операциями. Дополнительную информацию о выражениях см. в. разделе “Выражения” на стр. 231.

оператор_сравнения

Один из следующих:

Оператор	Значение
=	Равенство
>	Больше, чем
<	Меньше, чем
>=	Больше или равно

Оператор	Значение
<=	Меньше или равно
!=	Не равно
<>	Не равно
!>	Не больше, чем
!<	Не меньше, чем

При сравнении данных типов char, nchar, unichar, varchar, univarchar и nvarchar одно выражение меньше другого, если оно ближе к началу алфавита, и больше, если ближе к концу алфавита.

Сравнение регистра и специальных символов зависит от схемы упорядочения операционной системы на машине, где установлен Adaptive Server. Например, буквы нижнего регистра могут быть логически больше букв верхнего регистра, а буквы верхнего регистра – больше чисел.

При сравнении завершающие пробелы игнорируются. Например, “Dirk” – то же самое, что и “Dirk ”.

Когда сравниваются даты, более ранняя дата считается меньшей, чем более поздняя. Все символьные данные и даты в операторе сравнения необходимо заключать в кавычки. Например:

```
= "Bennet"
> "94609"
```

Правила ввода данных см. в разделе “Пользовательские типы данных”.

like

Ключевое слово, означающее, что следующая за ним строка символов (заключенная в одинарные или двойные кавычки) – шаблон соответствия. Ключевое слово like применимо к столбцам типа char, varchar, unichar, univarchar, nchar, nvarchar и datetime, но не предназначено для поиска значений секунд или миллисекунд.

Ключевое слово like и метасимволы можно использовать с данными типа datetime, а также с данными типов char и varchar. Когда ключевое слово like используется со значениями типа datetime, даты преобразуются в стандартный формат datetime, а затем – в varchar. Поскольку стандартный формат хранения не включает секунды и миллисекунды, нельзя искать значения секунд или миллисекунд с использованием like и шаблона.

Ключевое слово like удобно при поиске значений типа datetime, поскольку записи datetime могут содержать самые резные части дат. Например, при вставке значения “9:20” в столбец arrival_time следующая инструкция не найдет его, т.к. Adaptive Server преобразует эту запись в “Jan 1, 1900 9:20AM.”:

```
where arrival_time = '9:20'
```

Однако следующая инструкция найдет эту запись:

```
where arrival_time like '%9:20%'
```

строка_соответствия

Строка символов и метасимволов, заключенная в кавычки. Метасимволы перечислены в таблице 7-38.

Таблица 7-38. Метасимволы

Метасимвол	Значение
%	Любая строка, содержащая 0 и более символов
_	Любой одиночный символ
[ ]	Любой одиночный символ, входящий в указанный диапазон ([a-f]) или множество ([abcdef])
[^]	Любой одиночный символ, не входящий в указанный диапазон ([^a-f]) или множество ([^abcdef])

escape

Задает экранирующий символ, с помощью которого можно искать метасимволы.

экранирующий_символ

Любой одиночный символ. Дополнительную информацию см. в разделе “Использование инструкции escape” на стр. 254.

is null

Используется для поиска неопределенных значений.

between

Ключевое слово начала диапазона. Для указания конечного значения диапазона используется слово and. Следующий диапазон является включающим:

```
where @val between x and y
```

А этот диапазон – нет:

```
x and @val < y
```

Запросы, использующие слово between, не возвращают строк, если первое указанное значение больше второго.

and
: Соединяет два условия и возвращает результаты, если оба этих условия истинны.

  Если в команде используется более одного логического оператора, операторы and обычно оцениваются первыми. Впрочем, порядок выполнения операторов можно изменить с помощью круглых скобок.

in
: Позволяет выбирать значения, совпадающие с любым значением в списке. Компаратор может быть константой или именем столбца, а список может быть набором констант или, что более распространено, подзапросом. Информацию об использовании ключевого слова in с подзапросами см. в книге *Transact-SQL User's Guide*. Список значений должен быть заключен в круглые скобки.

список_значений
: Список значений. Символьные значения должны быть заключены в одинарные или двойные кавычки и разделены запятыми (см. пример 7). Список может быть списком переменных, например:

```
in (@a, @b, @c)
```

  Однако в список значений не может входить переменная, сама содержащая список, например, следующая:

```
@a = "'1', '2', '3'"
```

exists
: Используется с подзапросом для проверки того, возвращает ли этот подзапрос какой-либо результат. Дополнительную информацию см. в книге *Transact-SQL User's Guide*.

подзапрос
: Ограниченный оператор select (инструкции order by и compute, а также ключевое слово into недопустимы) внутри инструкции where или having оператора select, insert, delete или update либо подзапроса. Дополнительную информацию см. в книге *Transact-SQL User's Guide*.

any
: Используется с >, < или = и подзапросом. Возвращает результаты, если какое-либо значение, найденное в подзапросе, совпадает со значением в инструкции where или having внешнего оператора. Дополнительную информацию см. в книге *Transact-SQL User's Guide*.

all

Используется с > или < и подзапросом. Возвращает результаты, если все значения, найденные в подзапросе, совпадают со значением в инструкции where или having внешнего оператора. Дополнительную информацию см. в книге *Transact-SQL User's Guide*.

имя_столбца

Имя столбца, используемого в сравнении. Имя столбца должно уточняться именем его таблицы или представления, если возможна двусмысленность. Для столбцов со свойством IDENTITY можно указывать ключевое слово syb_identity, при необходимости уточненное именем таблицы, вместо фактического имени столбца.

оператор_соединения

Оператор сравнения или один из операторов соединения =* или *=. Дополнительную информацию см. в книге *Transact-SQL User's Guide*.

логическое_выражение

Выражение, возвращающее TRUE или FALSE.

or

Соединяет два условия и возвращает результаты при соблюдении одного из условий.

Если в команде используется более одного логического оператора, операторы or обычно оцениваются после операторов and. Впрочем, порядок выполнения операторов можно изменить с помощью круглых скобок.

Примеры

Пример 1.

```
where advance * $2 > total_sales * price
```

Пример 2. Поиск всех строк, в которых номер телефона начинается не с 415:

```
where phone not like '415%'
```

Пример 3. Поиск строк, в которых имена авторов – Carson, Carsen, Karsen и Karson:

```
where au_lname like "[CK]ars[eo]n"
```

Пример 4. Поиск строки таблицы sales_east, в которой значение столбца IDENTITY равно 4:

```
where sales_east.syb_identity = 4
```

Пример 5.

```
where advance < $5000 or advance is null
```

Пример 6.

```
where (type = "business" or type = "psychology") and advance > $5500
```

Пример 7.

```
where total_sales between 4095 and 12000
```

Пример 8. Поиск строк, в которых штат соответствует одному из трех значений в списке:

```
where state in ('CA', 'IN', 'MD')
```

Использование

- Условия поиска where и having идентичны за исключением того, что в инструкциях where недопустимо использование агрегатных функций. Например, следующая инструкция корректна:

  ```
  having avg(price) > 20
  ```

  А эта инструкция – нет:

  ```
  where avg(price) > 20
  ```

  Дополнительную информацию об использовании агрегатных функций и примеры см. в главе 2, “Функции Transact-SQL” и в разделе group by и having на стр. 576.

- Соединения и подзапросы указываются в условиях поиска: полное описание см. в книге *Transact-SQL User’s Guide*.

- Количество условий and и or в инструкции where ограничивается только объемом памяти, доступной для выполнения запроса.

- Строка шаблона, включаемая в предикат like, ограничивается только размером строки, которая может быть помещена в varchar.

- Существует два способа указания кавычек в записи типа char или varchar. Первый способ – использовать две кавычки. Например, если символьная запись начинается с одинарной кавычки и необходимо, чтобы одинарная кавычка являлась частью этой записи, необходимо использовать две одинарные кавычки:

  ```
  'I don''t understand.'
  ```

  Или использовать двойные кавычки:

  ```
  "He said, ""It's not really confusing."""
  ```

  Второй способ – заключать кавычку одного типа в кавычку другого типа. Другими словами, заключать запись с двойными кавычками в одинарные кавычки (или наоборот). Вот несколько примеров:

```
'George said, "There must be a better way."'
"Isn't there a better way?"
'George asked, "Isn"t there a better way?"'
```

- Для ввода символьной строки, которая длиннее ширины экрана, необходимо ввести обратную косую черту (\) перед переходом на следующую строку.
- При сравнении столбца с константой или переменной в инструкции where Adaptive Server преобразует константу или переменную в тип данных этого столбца, чтобы оптимизатор мог использовать индекс для поиска данных. Например, выражения типа float преобразуются в int при сравнении со столбцом типа int. Например:

  ```
  where int_column = 2
  ```

  выбирает строки, где *int_column* = 2.
- Когда Adaptive Server оптимизирует запросы, он оценивает условия поиска в инструкциях where и having и определяет, какие условия – аргументы поиска, которые можно использовать при выборе наилучших индексов и плана запроса. Для отбора строк используются все условия поиска. Дополнительную информацию об аргументах поиска см. в книге *Руководство по настройке производительности*.

Стандарты

Уровень соответствия стандарту SQL92: совместимость с базовым уровнем.

См. также

Команды – delete, execute, group by и having, insert, select, update

Типы данных – Типы данных для представления даты и времени

Системные процедуры – sp_helpjoins

while

Описание

Задает условие для повторного выполнения оператора или блока операторов. Операторы выполняются повторно, пока указанное условие истинно.

Синтаксис

while *логическое_выражение* [plan "*абстрактный план*"]
оператор

Параметры

логическое_выражение
Любое выражение, возвращающее TRUE, FALSE или NULL.

plan “абстрактный план”
Определяет абстрактный план для оптимизации запроса. Это может быть полный или частичный план, заданный на языке абстрактных планов. Планы можно задавать только для тех операторов SQL, которые поддаются оптимизации, то есть для запросов, обращающихся к таблицам. Дополнительную информацию см. в главе 30, “Руководство по написанию абстрактных планов”, книги *Руководство по настройке производительности*.

оператор
Может быть одним оператором SQL, но обычно это блок операторов SQL, разделенных ключевыми словами begin и end.

Примеры

Если средняя цена ниже 30 долларов, цены на все книги в таблице titles удваиваются. Пока средняя цена остается ниже 30 долларов, цикл while продолжает удваивать цены. Помимо определения названий книг, дороже 20 долларов, оператор select в цикле while показывает количество завершенных циклов (каждый результат средней цены, возвращаемый Adaptive Server, означает один цикл):

```
while (select avg(price) from titles) < $30
begin
    select title_id, price
        from titles
        where price > $20
    update titles
        set price = price * 2
end
```

Использование

- Выполнением операторов в цикле while можно управлять внутри цикла командами break и continue.
- Команда continue запускает цикл while с начала, пропуская все операторы после continue. Результатом выполнения команды break является выход из цикла while. Все операторы, расположенные после ключевого слова end, обозначающего конец цикла, выполняются. Команды break и continue часто активируются по условию if.

Например:

```
while (select avg(price) from titles) < $30
begin
        update titles
        set price = price * 2
    if (select max(price) from titles) > $50
        break
    else
        if (select avg(price) from titles) > $30
           continue
        print "Average price still under $30"
end

select title_id, price from titles
        where price > $30
```

Этот пакет продолжает удваивать цены всех книг в таблице titles, пока средняя цена книги остается ниже 30 долларов. Однако если цена любой книги превысит 50 долларов, команда break остановит цикл while. Команда continue не допускает выполнения оператора print, если средняя цена книги превышает 30 долларов. Независимо от того, как завершается цикл while (нормально или по команде break), последний запрос отображает список книг, дороже 30 долларов.

- Если два или более цикла while вложены друг в друга, команда break выходит в следующий внешний цикл. Выполняются все операторы, расположенные после внутреннего цикла, а затем начинается выполнение следующего внешнего цикла.

Предупреждение. Если команды create table или create view встречаются внутри цикла while, сервер Adaptive Server создает схему для таблицы или представления до определения истинности условия. Это может привести к ошибкам, если таблица или представление уже существуют.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду while по умолчанию могут выполнять все пользователи. Для ее выполнения никаких полномочий не требуется.

См. также

Команды – begin...end, break, continue, goto label

writetext

Описание

Разрешает минимально регистрируемое интерактивное обновление существующего столбца типа text или image.

Синтаксис

writetext [[*база_данных*.]*владелец*.]*имя_таблицы.имя_столбца*
указатель_на_данные [readpast] [with log] *данные*

Параметры

имя_таблицы.имя_столбца

Имя таблицы и обновляемого столбца типа text или image. Если таблица находится в другой базе данных, также нужно указать имя базы данных, а если в базе данных существует несколько таблиц с таким именем, то нужно также указать имя владельца. По умолчанию *владелец* – текущий пользователь, а *база_данных* – текущая база данных.

указатель_на_данные

Значение типа varbinary(16), в котором хранится указатель на данные типа text или image. Чтобы определить это значение используется функция textptr, как показано в примере 1. Данные типа text и image хранятся отдельно от других столбцов таблицы. (в отдельном наборе связанных страниц). Указатель на фактическое местоположение хранится вместе с данными; функция textptr возвращает этот указатель.

readpast

Дает указание команде модифицировать только незаблокированные строки. Если команда writetext находит заблокированные строки, она их пропускает, не дожидаясь снятия блокировки.

with log

Регистрирует в журнале вставленные данные типа text или image. Использование этого параметра полезно при восстановлении носителя, но запись в журнал больших блоков данных быстро увеличивает размер журнала транзакций, поэтому необходимо убедиться, что журнал транзакций находится на отдельном устройстве. Дополнительную информацию см. в описании команды create database, системной процедуры sp_logdevice и в книге *Руководство по системному администрированию*.

данные

Данные, записываемые в столбец типа text или image. Данные типа text необходимо заключать в кавычки. Данным типа image должно предшествовать “0x”. Проверьте информацию об используемом клиентском программном обеспечении, чтобы определить максимальную длину данных типа text или image, которые могут быть приняты клиентом.

Примеры

Пример 1. В этом примере указатель на текст помещается в локальную переменную *@val*. Затем команда writetext помещает текстовую строку “hello world” в текстовое поле, на которое указывает переменная *@val*:

```
declare @val varbinary(16)
select @val = textptr(copy) from blurbs
    where au_id = "409-56-7008"
writetext blurbs.copy @val with log "hello world"
```

Пример 2.

```
declare @val varbinary(16)
select @val = textptr(copy)
from blurbs readpast
    where au_id = "409-56-7008"
writetext blurbs.copy @val readpast with log "hello
world"
```

Использование

- Максимальный размер текста, который можно интерактивно вставлять в столбцы типа text и image с помощью команды writetext, приблизительно равен 120 Кб.
- По умолчанию операция writetext регистрируется по минимуму. В журнал заносится только выделение и освобождение страниц. Данные типа text или image при записи в базу данных не регистрируются. Чтобы использовать команду writetext в режиме минимальной журнализации (который установлен по умолчанию), системный администратор должен с помощью процедуры sp_dboption задать параметру select into/bulkcopy/pllsort значение true.
- Команда writetext обновляет данные типа text в существующей строке. В результате такого обновления весь существующий текст полностью заменяется новым.
- Операции writetext не отслеживаются триггером insert или update.
- Команда writetext требует действительного указателя на столбец типа text или image. Чтобы такой указатель существовал, столбец типа text должен содержать либо фактические данные, либо значение NULL, которое было явно введено командой update.

  Возьмем таблицу textnull со столбцами textid и *x*, где *x* – столбец типа text, допускающий неопределенные значения; команда update присваивает NULL всем значениям столбца x и определяет действительный указатель в столбце text:

  ```
  update textnull
  set x = null
  ```

Никакой указатель на текстовые данные или изображения не создается при вставке с помощью команды insert явного значения NULL:

```
insert textnull values (2,null)
```

Аналогично, никакой указатель на текстовые данные или изображения не создается при вставке с помощью команды insert неявного значения NULL:

```
insert textnull (textid)
values (2)
```

- Операции insert и update для столбцов типа text регистрируются в журнале.
- Нельзя использовать команду writetext для столбцов типа text и image в представлениях.
- После перехода на многобайтовый набор символов необходимо выполнить команду dbcc fix_text. Если этого не сделать, то попытка применить команду writetext к значениям типа text завершится сбоем и появится сообщение об ошибке с предложением выполнить dbcc fix_text для этой таблицы.
- Команда writetext в режиме по умолчанию, то есть без журнализации, выполняется медленнее на фоне работы оператора dump database.
- Функции библиотеки Client-Library dbwritetext и dbmoretext работают быстрее и используют меньше динамической памяти, чем команда writetext. Эти функции могут вставлять до 2 ГБ данных типа text.

Использование параметра *readpast*

- Параметр readpast применяется лишь к таблицам с блокировкой только данных. readpast игнорируется, если он указан для таблиц с блокировкой всех страниц.
- Если в сеансе уровень изоляции равен 3, параметр readpast игнорируется без уведомления.
- Если в сеансе уровень изоляции транзакций равен 0, команды writetext с параметром readpast не выдают предупреждающих сообщений. Они изменяют указанный текстовый столбец, если этот текстовый столбец не заблокирован несовместимыми блокировками.

Стандарты

Уровень соответствия стандарту SQL92: расширение Transact-SQL.

Полномочия

Команду writetext по умолчанию может выполнять владелец таблицы, который может передавать это полномочие другим пользователям.

См. также

Команды – readtext

Типы данных – Типы данных text и image