ПРОЕКТИРОВАНИЕ И СОЗДАНИЕ БАЗ ДАННЫХ

Методические указания к лабораторной работе №1

1. ЦЕЛЬ РАБОТЫ

Целью работы является приобретение практических навыков применения концептуальной модели «сущность – связь» для создания баз данных.

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СПРАВКА

2.1. Модель «сущность – связь».

Проектирование современных баз данных включает три этапа:

- концептуальное проектирование;
- логическое проектирование;
- физическое проектирование.

На каждом этапе используется соответствующая модель данных – концептуальная, логическая и физическая.

Модель «сущность – связь» относится к концептуальным моделям. В современном своем варианте она также является и логической моделью данных. Модель «сущность – связь» чаще всего применяется для проектирования структур баз данных.

Важной особенностью модели «сущность – связь» является то, что по ней однозначно может быть построена физическая модель базы данных или ее *схема*.

Поэтому практически все системы автоматизированного проектирования баз данных (CASE – системы) используют данную модель. В результате создан язык моделирования ERD (Entity-RelationshipDiagrams) – язык диаграмм «сущность – связь», применяемый в CASE – системах.

Первый вариант модели «сущность – связь» был предложен в 1976 г. Питером Ченом[1]. В дальнейшем многими авторами были разработаны свои варианты подобных моделей (нотация Мартина, нотация IDEF1X, нотация Баркера и др.). Кроме того, различные CASE – системы, реализующие одну и ту же нотацию, могут отличаться своими возможностями. По сути, все варианты диаграмм

«сущность – связь» исходят из одной идеи - использовать графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

Модель "сущность-связь" не является строго формальной. Диаграммы ERD строятся согласно следующим определениям.

Сущность - это класс однотипных объектов, информация о которых должна быть учтена в модели. Слово “ класс” не имеет здесь отношения к объектно – ориентированному программированию. Однако, существует понятие *подсущность*. Подсущность наследует свойства сущности, которой она принадлежит.

Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Примерами сущностей могут быть такие классы объектов как "Студент", "Адрес", "Человек". Каждая сущность в модели изображается в виде прямоугольника с наименованием.

Экземпляр сущности - это конкретный представитель данной сущности. Например, представителем сущности " Человек " может быть "Иванов". Экземпляры сущностей должны быть различимы, т.е. сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности.

Атрибут сущности - это именованная характеристика, являющаяся некоторым свойством (параметром) сущности. Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными). Примерами атрибутов сущности "Человек" могут быть такие атрибуты как "Фамилия", "Имя", "Отчество", "Номер паспорта" и т.п. Атрибуты изображаются в пределах прямоугольника, определяющего сущность.

На рис. 1. показан пример диаграммы сущности, построенной в CASE – системе. При задании сущности можно сразу определить тип данных, если он уже известен, и обязательный (mandatory – M) или необязательный статус данных конкретного атрибута.

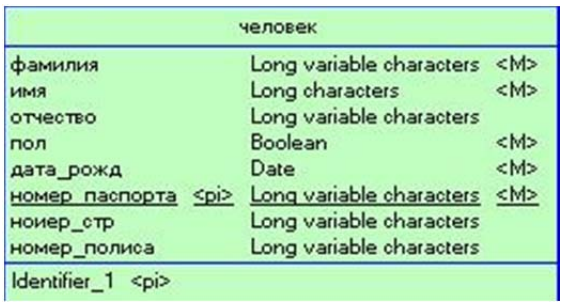

Рис. 1. Пример диаграммы сущности «Человек» с указанием атрибутов, типов данных и их обязательности.

Ключ сущности - это неизбыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Неизбыточность заключается в том, что удаление любого атрибута из ключа нарушается его уникальность. Сущность может иметь несколько различных ключей. Ключевые атрибуты изображаются на диаграмме подчеркиванием, как атрибут «Номер паспорта» на рис. 1.

Связь - это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Графически связь изображается линией, соединяющей две сущности:

Каждая связь имеет собственное имя, два конца и два наименования. Наименование обычно выражается в неопределенной глагольной форме: "иметь", "принадлежать" и т.п. Каждое из наименований относится к своему концу связи. Наименования можно не указывать. Будучи объявленными, имя связи и наименования концов обрабатываются CASE – системой и применяются в алгоритмах обработки диаграмм “сущность - связь” для построения таблиц базы данных.

Каждая связь имеет два параметра: *тип* и *модальность*. Примеры графического отображения типов связей показаны на рис. 2.

Рис. 2. Примеры типов связей диаграмм “сущность - связь”.

Связь типа *один-к-одному* означает, что один экземпляр первой сущности связан только с одним экземпляром второй сущности. Связь один-к-одному может соответствовать ситуации, когда две сущности отражают разные характеристики одного и того же объекта.

Связь типа *один-ко-многим* означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны "один") называется родительской, правая (со стороны "много") - дочерней. Характерный пример такой связи приведен на рис. 4.

Связь типа *много-ко-многим* означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и

каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.

Каждая связь может иметь одну из двух модальностей связи :

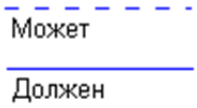

Рис. 3. Модальности связей диаграмм “сущность - связь”.

Модальность "может" означает, что экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может и не быть связан ни с одним экземпляром.

Модальность "должен" означает, что экземпляр одной сущности обязан быть связан не менее чем с одним экземпляром другой сущности.

Связь может иметь разную модальность с разных концов (как на рис. 4.).

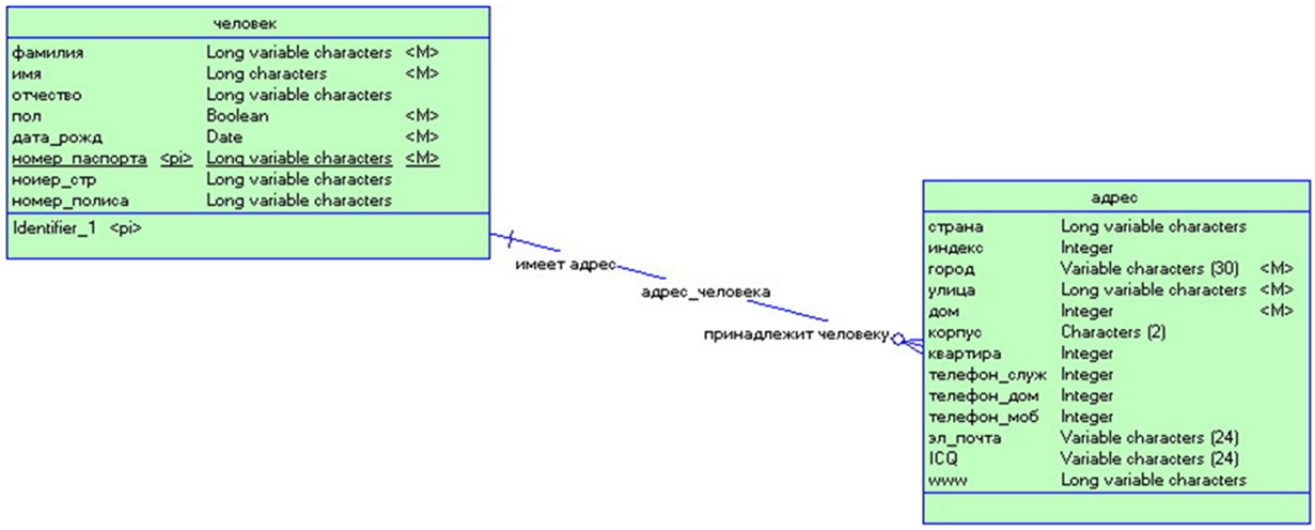

Рис. 4. Пример связанных сущностей.

2.2. Физическая модель базы данных

Физическая модель базы данных – это совокупность проектных решений, в которых определяется следующее:

- платформа СУБД - формат данных базы, особенности реализации операторов языка SQL;

- имена таблиц, имена полей, типы данных полей;
- ключи для связывания таблиц – *первичные*, *внешние*, их имена и состав;
- иные объекты, включаемые в состав базы данных, например, *виды*, *триггеры*, *процедуры*.

Большинство CASE – систем способны преобразовывать модели "сущность-связь" в физические модели баз данных автоматически. При этом различаются CASE – системы степенью автоматизации такого преобразования, широтой охвата современных платформ и гибкостью настроек параметров алгоритмов преобразования.

Поскольку во всех реляционных базах данных в качестве стандартного языка управления данными применяется язык SQL, физическая модель базы данных реализуется в виде команд языка SQL, образующих специальный набор – т.н. *скрипт*. Каждый скрипт представляет собой последовательность команд SQL, определяющих создание объектов базы данных – *таблиц, ключей*, *видов*, *триггеров*, *процедур,* а также команд на заполнение таблиц данными. Последнее имеет место, когда концептуальная модель *достаточно подробна* и содержит определение экземпляров данных, реализующих сущности.

Концептуальные модели различаются степенью проработки, но в любом случае CASE – системы преобразуют их в систему SQL – скриптов, которые также более или менее детально описывают создание базы данных. Поэтому процесс проектирования и создания базы данных может быть итерационным: получив скрипты, выполнив их анализ, можно вернуться к исходным моделям, изменить или дополнить их и вновь сгенерировать скрипты физической модели базы данных. В этом состоит суть построения физических моделей баз данных.

2.3. Построения физических моделей баз данных

Модель "сущность-связь" можно преобразовать в систему отношений, пользуясь алгоритмом, известным как *правила Джексона* [4]. Этот алгоритм является эвристическим, но для диаграмм "сущность-связь", содержащих небольшое число сущностей, как правило, не более 20–ти, он дает хорошие результаты.

Алгоритм представляет собой шесть правил, которые используют следующие параметры диаграмм ERD:

- степень связи;
- модальность связи;
- ключи сущностей.

Согласно этим правилам, сущности и связи диаграмм ERD превращаются в систему связанных отношений. Строго говоря, эти отношения еще не являются

таблицами базы данных. Если система отношений не нормализована, то ее нормализация может привести к большему числу таблиц, чем число отношений в системе.

Но на практике часто используют правила Джексона так, что считают результаты их применения готовыми таблицами базы данных. Вопрос о нормализации такой базы, очевидно, остается открытым.

В свою очередь, современные CASE – системы, по-крайней мере, согласно их рекламным материалам, гарантируют нормализацию генерируемых ими реляционных баз данных вплоть до третьей нормальной формы. Приведем правила Джексона так, как они изложены автором в книге [4]

Правило 1

Если степень связи равна 1:1 и модальности обеих сущностей являются обязательными, то требуется только одно отношение. Ключом этого отношения может быть ключ любой из двух сущностей.

Данное правило описывает ситуацию, когда обе сущности задают один и тот же объект, отражая разные его атрибуты. Например, это могут быть персональные данные человека – *фамилия, имя, отчество*, - и его биометрические данные – *рост, вес, группа крови*. Согласно данному правилу, все эти данные можно хранить в одной таблице. Тем не менее, вопрос о числе таблиц для диаграмм рассматриваемого здесь вида может решаться не столь однозначно. Если предполагается, что рассматриваемые здесь сущности могут быть связаны не только между собой, но и с другими сущностями, то, возможно, каждую из них следует реализовать в виде отдельной таблицы.

Правило 2.

Если степень связи равна 1:1 и модальность одной сущности является обязательной, а другой - необязательной, то необходимо построение двух отношений. Под каждую сущность необходимо выделение одного отношения, при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме того, ключ сущности, для которого модальность является необязательной, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательноймодальностью.

Логика данного правила такова: если модальность сущности необязательна, то есть она не всегда может иметь связь с другой сущностью, то хранить в ее таблице ссылку - внешний ключ от другой сущности нельзя, поскольку его поле может оказаться пустым. Внешний ключ – ссылка должен быть в таблице для сущности с обязательноймодальностью,и тогда он не будет пустым. Здесь действует некоторый принцип компактности.

Пример: сущность *Человек* с необязательноймодальностьюк связи с сущностью *Адрес*, у которой модальность обязательна: если адрес задан, то он обязательно кому-то принадлежит.

Правило 3.

Если степень связи равна 1:1 и модальность ни одной сущности не является обязательныой, то необходимо использовать три отношения: по одному для каждой сущности, ключи которых служат в качестве первичных в соответствующих отношениях, и одного для связи. Среди своих атрибутов отношение, выделяемое связи, будет иметь по одному ключу сущности от каждой сущности.

Здесь та же логика, что и в предыдущем правиле. Ее применение делает необходимым наличие трех таблиц – отношений. В служебной таблице, моделирующей связь, имеется, по-крайней мере, два поля – это внешние ключи – ссылки от сущностей; они никогда не пусты.

Правило 4.

Если степень связи равна 1: n и модальность n-связной сущности является обязательной, то достаточным является использование двух отношений, по одному на каждую сущность, при условии, что ключ каждой сущности служит в качестве первичного ключа для соответствующего отношения. Дополнительно ключ 1-связной сущности должен быть добавлен как атрибут в отношение, отводимое n-связной сущности.

Связь типа 1: *n* означает, что с каждым экземпляром одной сущности могут быть связаны несколько экземпляров другой сущности. Если поместить внешний ключ – ссылку в таблицу для односвязной сущности, то для всех экземпляров *n*-связной сущности придется продублировать данные, находящиеся в таблице для 1-связной сущности, что расточительно. Но самое главное в том, что значение ключа может оказаться пустым при необязательном классе принадлежности 1-связной сущности. Если применить правило, то в таблице для *n*-связной сущности будут продублировано только значение ключа-ссылки на 1-связную сущность. Принцип компактности действует и здесь!

Правило 5.

Если степень связи равна 1:n и модальность n-связной сущности является необязательным, то необходимо формирование трех отношений: по одному для каждой сущности, причем ключ каждой сущности служит первичным ключом соответствующего отношения, и одного отношения для связи. Связь должна иметь среди своих атрибутов ключ сущности от каждой сущности.

Здесь необходимо иметь три таблицы-отношения, чтобы исключить пустые значения в полях внешних ключей в таблицах исходных сущностей.

Правило 6.

Если степень связи равна m:n, то для хранения данных необходимо три отношения: по одному для каждой сущности, причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения, и одного отношения для связи. Последнее отношение должно иметь в числе своих атрибутов ключ – ссылку от каждой сущности.

Данное правило абсолютно: для пары «многосвязных» сущностей всегда создаются три таблицы – отношения.

Дополнительные замечания к правилам.

1. Правила Джексона применяются к парам сущностей диаграмм ERD, связанных бинарными связями. На практике возможны более сложные конфигурации связей, например, циклические связи. Все эти варианты требуют дополнительных исследований.

2. Формально две сущности могут быть связаны более чем одной связью. При этом возникает проблема приоритета связей при применении правил Джексона.

2.4. Реализация физических моделей баз данных

Как уже отмечалось, диаграмма "сущность-связь" может быть преобразована в физическую модель базы данных автоматически при помощи CASE- системы. Для диаграмм с небольшим числом сущностей это можно сделать вручную, применяя алгоритм (правила) Джексона.

Физическая модель базы данных должна быть реализована на конкретной платформе СУБД. Рассмотрим ее реализацию на платформе SAP-Sybase [5].

Базы данных современных платформ СУБД создаются и поддерживаются при помощи *инструментальных средств*. В платформе SAP-Sybase такими средствами являются система администрирования баз данных SybaseCentralи система программирования СУБД SybasePowerBuilder [5].

Рассмотрим реализацию физических моделей баз данных в системе PowerBuilder.

Базы данных – это комплексные объекты, управление которыми требует специальных программных решений. В системе PowerBuilder для этих целей служит *мастерская баз данных*.

2.4.1 Мастерская баз данных

Мастерская баз данных – это подсистема PowerBuilder, предназначенная для создания и редактирования баз данных формата *.db, являющегося одним из форматов баз данных платформы SAP-Sybase. Работая в системе PowerBuilder, программист –разработчик информационных систем может использовать несколько мастерских и оперативно переключаться между ними.
На рис. 5 показано главное окно мастерской баз данных PowerBuilder и приведены необходимые пояснения.

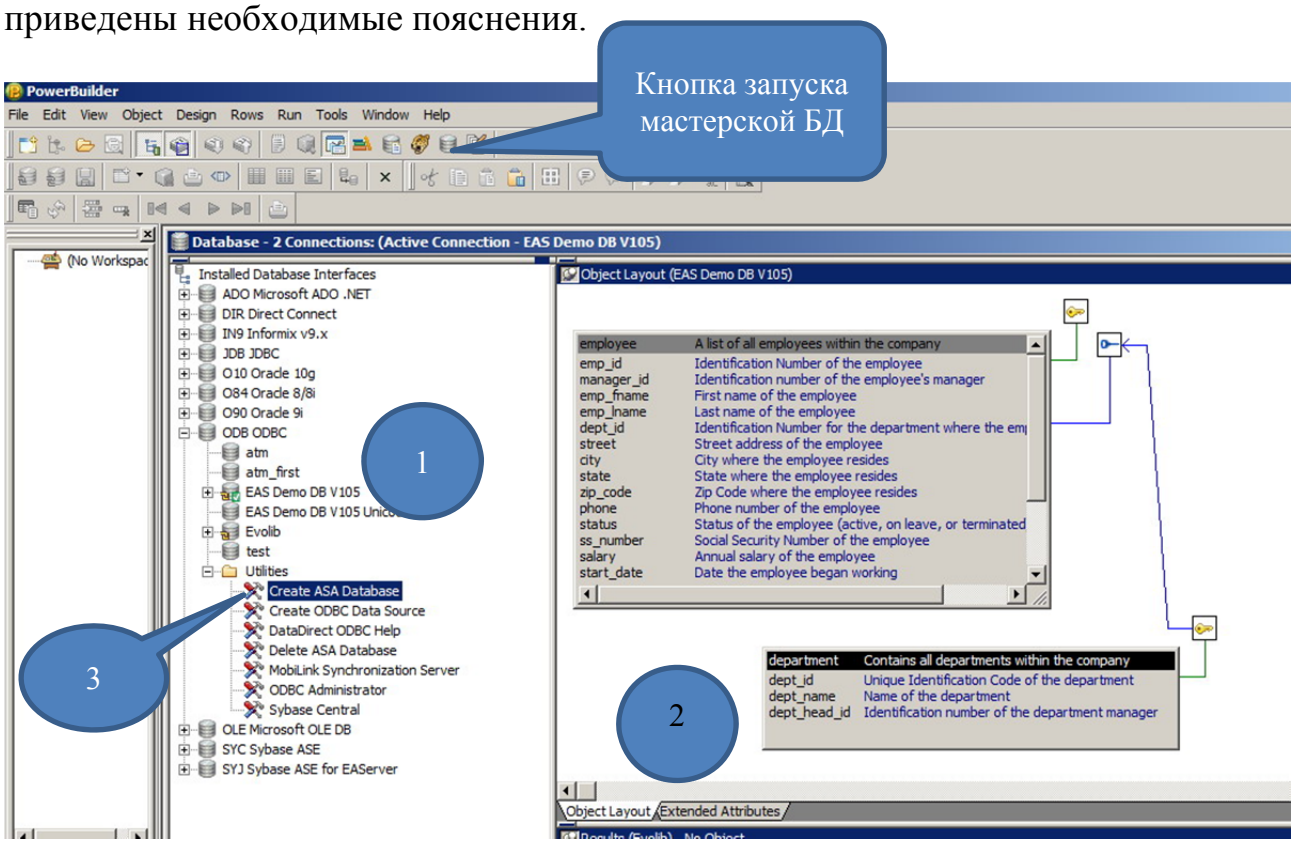

Рис. 5. Главное окно мастерской баз данных PowerBuilder. 1 – интерфейсы и профили соединения; 2 – связанные таблицы базы данных; 3 – вызов утилиты создания базы данных.

2.4.2. Создание базы данных.

Для создания «пустой» базы данных необходимо воспользоваться утилитой CreateASADatabase. Как вызвать данную утилиту показано на рис. 5.

Диалоговое окно утилиты показано на рис. 6. При создании базы данных необходимо указать следующие параметры:

ASAVersion – версия сервера баз данных; выбирается из списка установленных серверов;

UserID – имя пользователя базы данных;

Password - пароль пользователя базы данных;

имя и пароль пользователябазы данных устанавливаются по умолчанию;

DatabaseName – имя файла базы данных; здесь необходимо выбрать маршрут и имя файла;

UseTransactionLog - флажок – индикатор использования контрольного файла транзакций; следует убрать данный флажок, если планируется копировать файл базы данных на сменные носители;

PageSize – размер страницы памяти, выделяемой для базы данных; необходимо выбрать минимальный размер 1024 байта, если работаем на персональном компьютере.

Остальные параметры диалогового окна на рис. 6 можно не задавать.

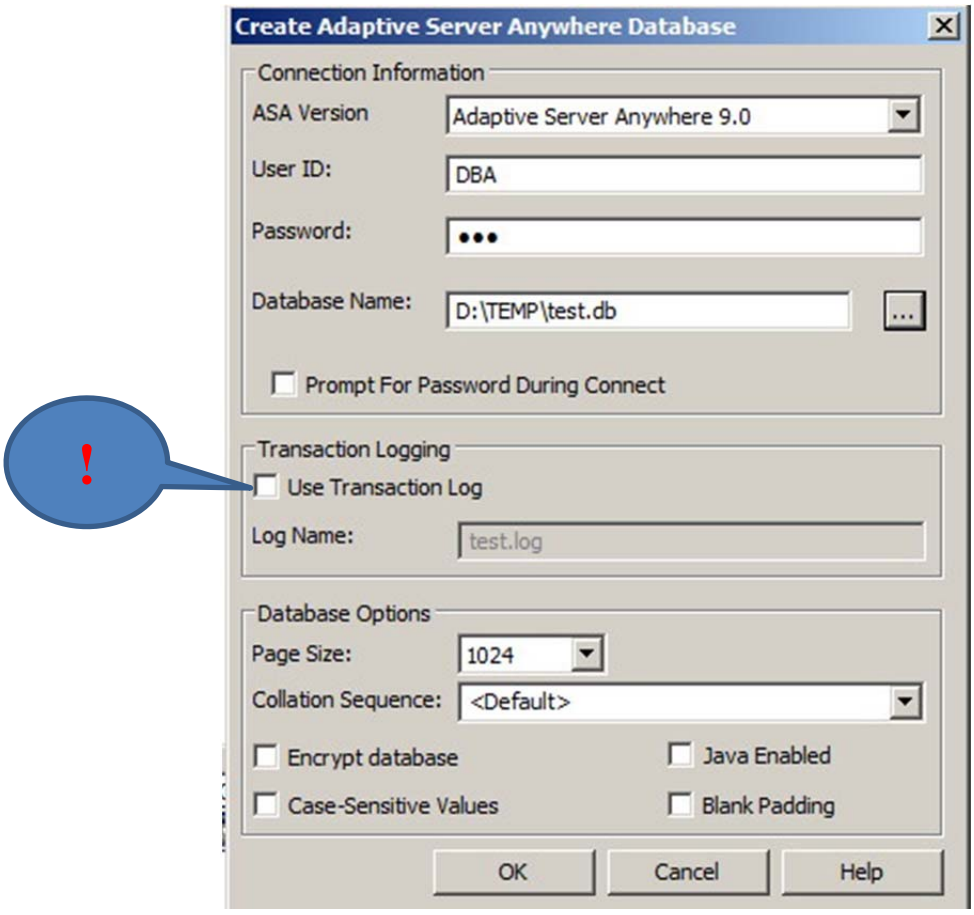

Рис. 6. Диалоговое окно утилиты CreateASADatabase.

После задания параметров в диалоговом окне, показанном на рис. 6, инициирование кнопки ОК приводит к генерации файла с именем <*имя*>.dbпо заданному маршруту, к созданию *источника данных*ODBCс именем <*имя*> и к созданию *профиля подключения* к базе данных с таким же именем.

Источник данных ODBC – это объект подсистемы Windows, предназначенной для соединения с базами данных различных платформ.Источник данных представляет собой настройки соответствующего драйвера для работы с сервером базы данных, который должен быть установлен в среде Windows.

Профиль подключения к базе данных - это объект системы PowerBuilder, представляющий собой также настройки для управления т.н. транзакциями; профиль необходим для работы с базами данных только в системе PowerBuilder.

Созданный профиль, в данном случае с именем test, показан на рис. 7.

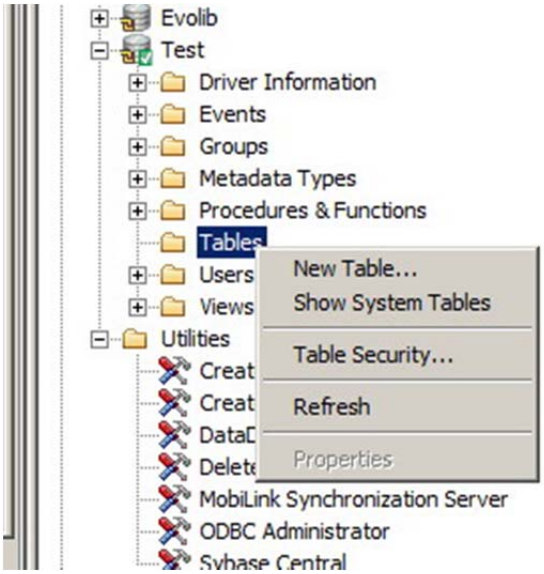

Рис. 7. Содержание профиля подключения к базе данных

Как видно из рис. 7, созданная вновь «пустая» база данных совсем не пуста: ее профиль содержит несколько папок, раскрыв которые можно увидеть их содержимое.Выбрав тему ShowSystemTablesв контекстном меню, всплывающем по нажатию правой кнопки мыши, позиционированной на папке Tables, можно увидеть значительное число системных таблиц, созданных в новой базе данных.

2.4.3. Создание таблиц.

В нашей отнюдь не пустой базе данных, тем не менее, нет «прикладных» таблиц, хранящих необходимые данные разрабатываемой СУБД. Эти таблицы должны пополнить папку Tablesпрофиля созданной базы данных. Для создания новой таблицы в контекстном меню, показанном на рис. 7, следует выбратьтему NewTable…, что приводит к появлению шаблона задания параметров таблицы, показанном на рис. 8.

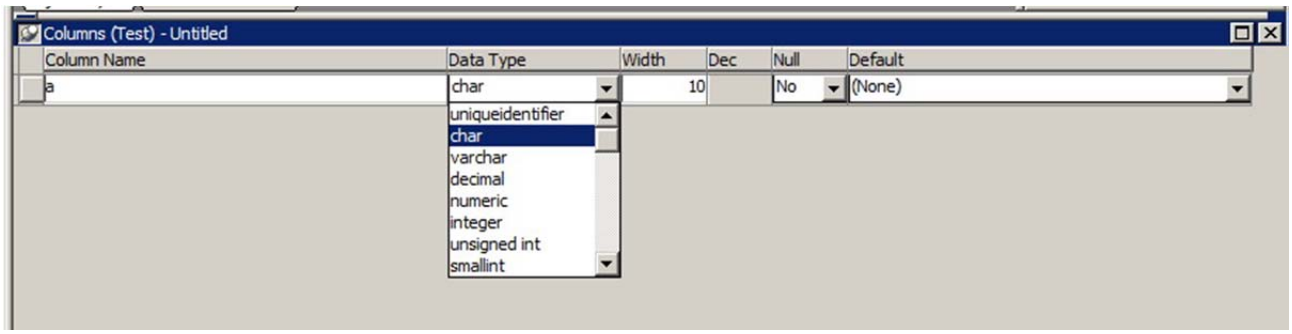

Рис. 8. Шаблон задания параметров таблицы

Используя данный шаблон, задаются имена полей таблицы (**ColumnName**), типы данных (**DataType**), ширина каждого поля (**Width**) – в зависимости от типа данных, при необходимости – число знаков после запятой (**Dec**), ограничение на пустые поля (**Null**), а также значение поля по умолчанию.

После определения всех полей содаваемой таблицы необходимо сохранить ее, выбрав тему Save…в системном меню PowerBuilder. При сохранении таблицы необходимо задать ее имя; в системе определяется владелец сохраняемой таблицы, которым по умолчанию является подключенный пользователь базы данных.

Созданная таблица отображается в папке Tables профиля подключения и может быть отображена в графическом режиме в окне ObjectLayoutмастерской баз данных, как показано на рис. 9.

2.4.4. Создание ключей.

Каждая таблица должна иметь *первичный ключ*. Для связывания таблиц применяются *внешние ключи*.Когда связываются две таблицы, одна из них является главной – *родительской*, а другая – подчиненной, *дочерней*. Первичный ключ служит для того, чтобы различать записи в таблице. Данные первичного ключа должны быть уникальны для каждой записи и не могут отсутствовать. Внешний ключ – это образ первичного ключа в дочерней таблице. Внешний ключ должен иметь такой же тип данных, что и первичный ключ, но его данные не обязательно уникальны.

Для создания первичного ключа удобно воспользоваться графическим изображением таблицы и правой кнопкой мыши инициировать появление контекстного меню, показанного на рис. 9.

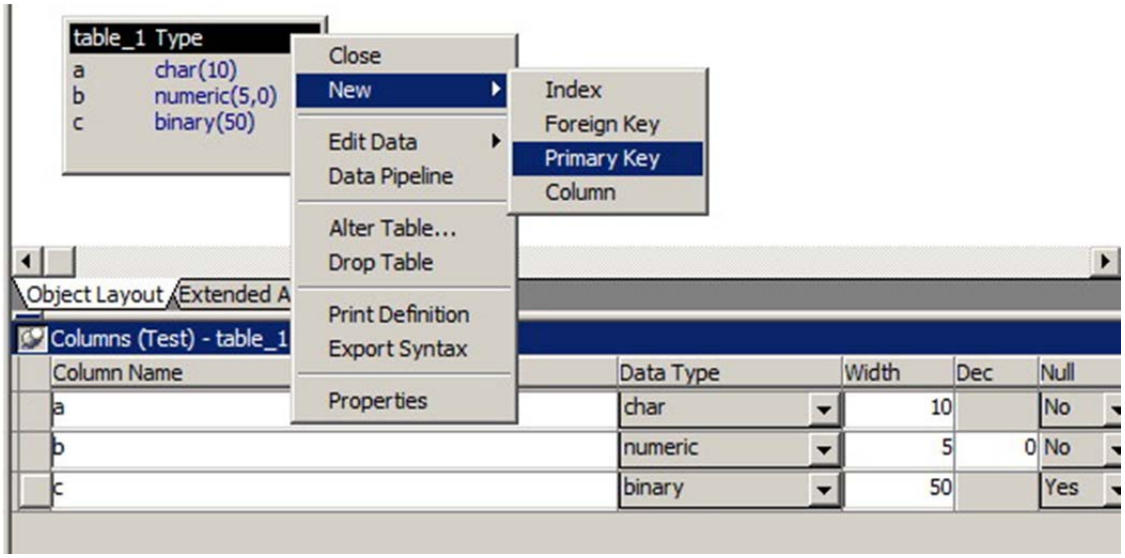

Рис. 9. Каскадное меню построения новых ключей, индексов и полей таблиц.

В этом каскадном меню необходимо выбрать тему PrimaryKeyи далее в находящемся справа окне параметров выбрать одно или несколько полей, составляющих первичный ключ.

Сохранив сделанные изменения в структуре таблицы, получим ее изображение, дополненное изображением ключа желтого цвета, соединенного зеленой линией с одним или несколькими полями таблицы, как показано на рис. 5.

Для создания внешнего ключа в дочерней таблице используем в ней то же меню на рис. 9, но выбираем тему ForeignKey.В окне свойств таблицы на рис. 10 назначаем имя внешнему ключу, выбираем его поле(я) и указываем на первичный ключ в родительской таблице. Для этого в закладке PrimaryKeyв окне свойств в выпадающем списке находим имя родительской таблицы - table_1на рис. 10.

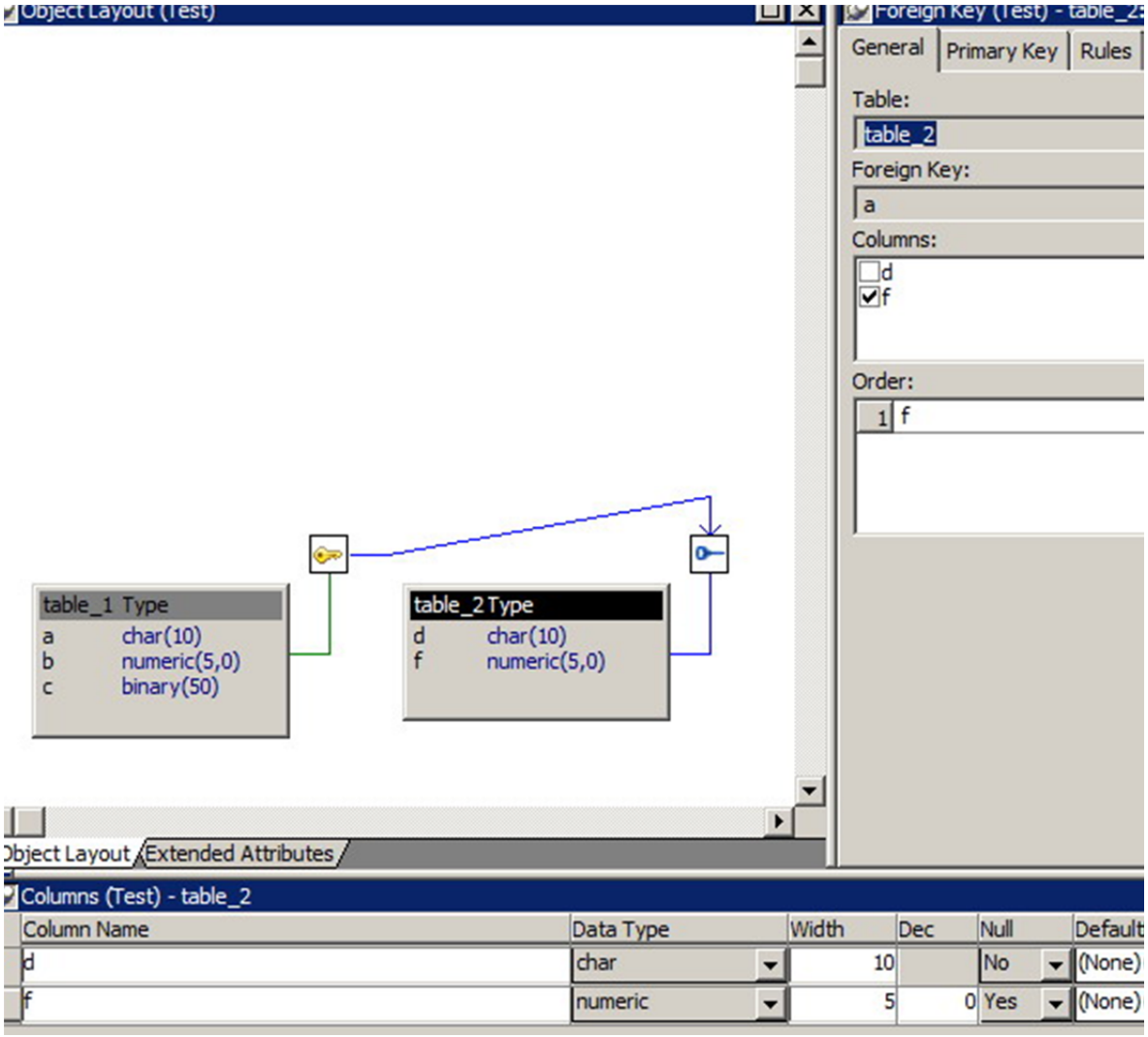

Рис. 10. Связанные по ключам таблицы базы данных.

Сделав все необходимые назначения, сохраняем результаты. Если все сделано правильно, то изображения связанных таблиц дополнятся изображениями ключей, как показано на рис. 10.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

3.1. Используя литературные источники, приведенные в списке литературы, изучить:

- принципы построения реляционных баз данных на основе концептуальных моделей [4];
- приемы работы в мастерской баз данных системы PowerBuilder [3], [5].

3.2. Получить задание на проектирование базы данных, отражающей информационное содержание заданных сущностей. Построить соответствующую заданию модель «сущность – связь».

3.3. По модели «сущность – связь» построить физическую модель базы данных.

3.4. Реализовать физическую модель базы данных в среде PowerBuilder.

3.5. Подготовить ответы на контрольные вопросы и выполнить контрольные упражнения.

4. СОДЕРЖАНИЕ ОТЧЕТА

Отчет по работе должен содержать:

- задание на проектирование базы данных;
- модель «сущность – связь», соответствующую заданию;
- алгоритм построения физической модели базы данных;
- физическую модель базы данных;
- объяснения к физической модели, касающиеся ограничений целостности созданной базы данных;
- реализацию физической модели в виде связанных таблиц базы данных.

Контрольные вопросы и упражнения

1. Классифицируйте модели данных, применяемых в СУБД.
2. Дайте определение понятию «Концептуальная модель данных»
3. Можно ли программировать, используя язык диаграмм?
4. Что такое CASE - система?
5. Связаны ли между собой модель данных и архитектура информационной системы?
6. Что входит в состав модели «сущность-связь»?
7. Почему CASE – система способна строить базы данных для любых платформ реляционных СУБД?
8. Что такое скрипт?
9. Для чего служит модель данных?
10. Что такое платформа СУБД?
11. Влияет ли архитектура технологии “клиент - сервер” на выбор модели базы данных?
12. Должно ли соответствовать имя профиля имени файла базы данных?
13. Должно ли соответствовать имя источника данных имени файла базы данных?
14. Какие ограничения вводятся при создании таблиц баз данных?
15. Может ли таблица базы данных иметь несколько внешних ключей?
16. Может ли таблица базы данных иметь несколько первичных ключей?
17. Могут ли внешние ключи быть «пустыми», т.е. не содержать данных?

Литература

1. Чен П. Модель "сущность-связь" - шаг к единому представлению о данных //СУБД. - 1995. - №3. - С.137-158.
2. Богатырев М.Ю. Разработка и программирование систем управления базами данных. - Тула, изд-во ТулГУ, 2009. - 145 с.
3. Богатырев М.Ю. Введение в систему PowerBuilder. Методические указания к выполнению лабораторных работ. - Тула, изд-во ТулГУ, 1998. - 36 с.
4. Джексон Г. Проектирование реляционных баз данных для использования с микро-ЭВМ. - М.: Мир, 1991. - 252 с.
5. СмитБ. Дж., ШаадГ.У. Power Builder 5.0. Библия разработчика. - К.: Диалектика, 1997. - 544 с.
6. Электронный ресурс: http://lis.tula.ru/Data/metodichka.rar

Приложение.
Варианты заданий в виде информационных сущностей

	Сущность
1.	автомобиль, владелец, страховая фирма
2.	студент, специальность, дисциплина
3.	собака, порода, хозяин
4.	футбольный клуб, игрок, матч
5.	компания, адрес, отрасль.
6.	актер, роль, спектакль
7.	самолет, пассажир, рейс
8.	Город, район, область
9.	кафедра, факультет, ВУЗ
10.	картина, художник, владелец
11.	автомобиль, запчасть, производитель
12.	звезда, созвездие, миф
13.	Университет, кафедра, сотрудник.
14.	Рассказ, Автор, Книга.
15.	Город, Район, Житель.
16.	Магазин, Отдел, Книга.
17.	Клиент, Банк, Филиал.
18.	Игра, жанр, страна
19.	Актер, фильм, страна
20.	Программа, компания-производитель, страна

21.	Товар, поставщик, город
22.	Работник, отдел, компания
23.	Статья, журнал, страна
24.	Пациент, отделение, больница
25.	Студент, группа, факультет
26.	Водитель, автомобиль, регион