Технология окон данных (Data Window)

Окно данных (DataWindow) – одно из основных средств работы в PowerBuilder. Работая с окнами данных, разработчик приложения определяет, как будет представлена на экране информация для конечного пользователя. Будет ли это окно запроса, форма для ввода данных или отчет для вывода на печать – в любом случае приходится иметь дело с окнами данных.

При выполнении приложения PowerBuilder окна данных никогда не существуют сами по себе, а всегда находятся в составе стандартного окна PowerBuilder.

При конструировании окна данных приходится иметь дело с тремя основными компонентами:

1. Объект DataWindow, создаваемый в мастерской окон данных (DataWindow Painter) и встраиваемый в элемент управления DataWindow в окне PowerBuilder.
2. Окно, разрабатываемое в мастерской окон (Window Painter). В нем располагаются все остальные объекты и элементы управления.
3. Элемент управления DataWindow – контроллер окна данных, размещаемый в окне и содержащий объект DataWindow. Элемент управления DataWindow ничем не отличается от других элементов управления окон, например от элемента CommandButton.

Для создания и использования окон данных необходимо иметь, по меньшей мере, следующие компоненты:

1. **Приложение**:
   - объект приложения;
   - окно;
   - сценарий открытия приложения, который открывал бы окно приложения;
   - файл библиотеки (.pbl), в котором хранится вся информация о приложении.
2. **Объект DataWindow**.
3. **Элемент управления (контроллер) DataWindow**.

Команды PowerScript. Используются для создания и управления базой данных и элементом управления DataWindow. Эти команды содержатся в сценариях событий для приложений и окон.

- Команда CONNECT и другие связанные с ней команды используются для подключения к базе данных. Обычно команда CONNECT выполняется из сценариев открытия приложений.
- Команда DISCONNECT используется для отключения от базы данных после завершения работы с ней. Обычно команда DISCONNECT выполняется из сценариев закрытия приложений.
- Функция SetTransObject используется для назначения элементу управления DataWindow объекта транзакции (обычно sqlca). Эта функция часто выполняется из сценариев открытия окна или из сценария создания элемента управления DataWindow.
- Функция Retrieve (Возвратить) используется для поиска и считывания строк из базы данных и отображения их в окне DataWindow. Обычно функцию Retrieve включают в состав сценария открытия окна, создания элемента управления DataWindow, щелчка на кнопке или вызова команды из меню.
- Функция Update используется для записи измененных строк в базу данных, если из окна, с которым вы работаете, разрешено производить такие изменения. Данную функцию можно использовать, например, в составе сценария закрытия окна, для удаления элемента управления DataWindow, в составе сценария щелчка на кнопке или элемента меню.

Для создания объекта DataWindow требуется запустить мастер окон данных (DataWindow Painter).

Вызова мастера окон данных:

- нажать кнопку New панели инструментов,
- выбрать закладку DataWindow.

Рис. 3.15. иллюстрирует данные действия.

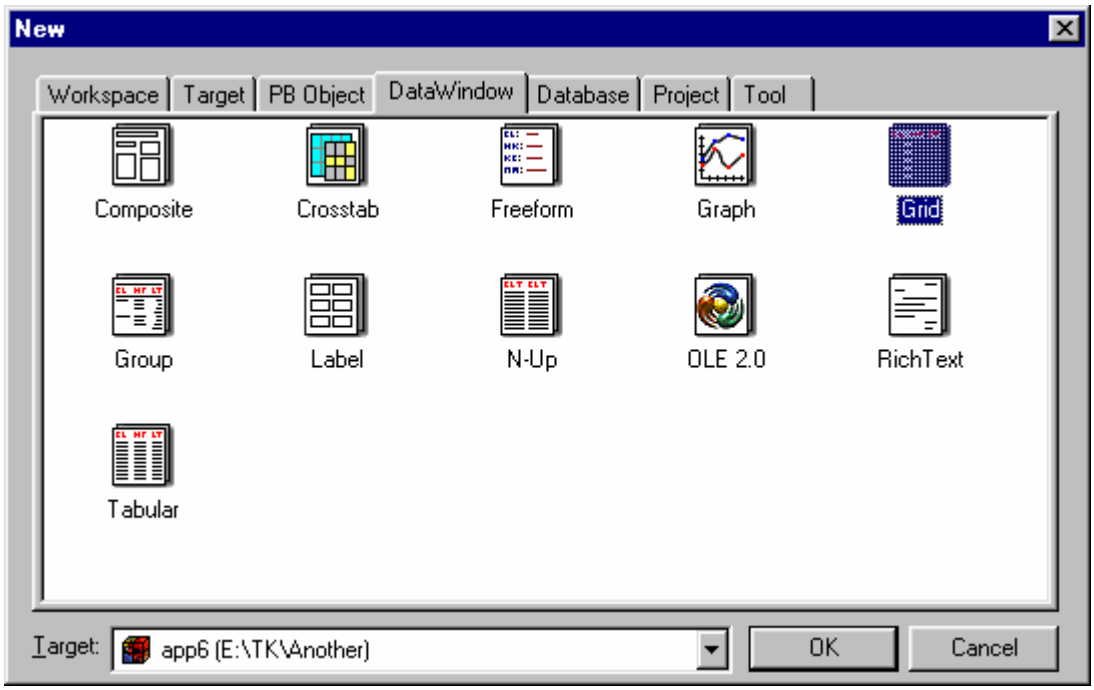

Рис. 3.15. Выбор типа окна данных.

Для того, чтобы разработать новое окно DataWindow, PowerBuilder попросит указать один из стилей представления (Presentation Style) и один из источников данных (Data Source). По типу источников данных определяется способ получения данных для окна, а по типу стиля представления – каким образом эти данные должны быть представлены в окне.

Предусмотрено 11 стилей представления:

1. Grid (Сетка). Используя этот стиль, PowerBuilder помещает данные в матрицу с ячейками, что довольно удобно, но не очень привлекательно.
2. FreeForm (Свободный стиль). В стиле FreeForm PowerBuilder располагает столбцы с данными вертикально, позволяя пользователю перегруппировать их на свое усмотрение. Поскольку в этом стиле данные и подписи можно поместить в произвольном порядке, он широко используется для самых разнообразных целей, в том числе для создания формальных писем и конвертов.
3. Tabular (Таблица). Так же, как и в стиле Grid, данные в окне DataWindow представлены в виде таблиц. Этот стиль более гибкий по сравнению со стилем Grid и дает возможность значительно улучшить внешний вид данных. Стиль Tabular очень часто используется при оформлении отчетов. Кроме того, он хорош для ввода данных в окно DataWindow.
4. Group (Группа). Очень похож на стиль Tabular, но дополнительно позволяет автоматически добавлять заголовок, номера страниц, итоговые и промежуточные значения данных.
5. Label (Почтовая наклейка). Позволяет создавать почтовые наклейки.

6. N-Up (N-колоночная разбивка). Разновидность таблицы, используется для более компактного отображения данных на странице, располагая данные в две или несколько колонок.
7. Graph (Диаграмма). Позволяет представить числовые данные в графическом виде.
8. CrossTab (Сводная таблица). Позволяет отобразить итоговые числовые данные в матрице.
9. Composite (Комбинированный стиль). В этом стиле в одном окне можно расположить несколько окон данных DataWindows. Стиль Composite очень удобен для создания отчетов по управлению и его можно использовать для ввода и отображения данных в окне DataWindow.
10. Rich Text (Текст в расширенном формате). С помощью стиля Rich Text можно создавать письма и другие документы, заполняя шаблон форматированного главного окна DataWindow данными из базы. Этот стиль позволяет достаточно широко использовать возможности форматирования, ориентированные на обработку текстов, например, верхние и нижние колонтитулы, а также многочисленные шрифты.
11. OLE 2.0. (OLE – связывание и внедрение объектов). При использовании процесса, который также называют “передача однородных данных”, информация из поддерживаемых PowerBuilder источников может быть передана на сервер-приложение OLE 2.0. Приложение-клиент применяет эти данные для создания диаграмм, схем, электронных таблиц и т.д., отображаемых в окне данных DataWindow.

Термин источник данных здесь отличается от введенного ранее, где он представлял собой имя связи базы данных ODBC. В PowerBuilder этот термин имеет двоякий смысл.

В DataWindow “источник данных” определяет, каким образом DataWindow получает данные. Существует пять вариантов, первые три из которых используют запрос SQL SELECT, а два других – описывают источники типа non-SELECT.

1. Quick Select (Быстрый выбор). Основываясь на сделанных разработчиком приложения настройках, PowerBuilder генерирует оператор SQL SELECT, который можно редактировать в мастере Select Painter.
2. SQL Select (Выбор SQL-запроса). Выбор пиктограммы SQL Select приведет непосредственно к вызову мастера Select Painter, где можно построить SQL-запрос до вызова мастера окон данных DataWindow Painter. Конечный результат при выборе как Quick Select, так и SQL Select – одинаковый, а именно SQL-запрос SELECT.
3. Query (Запрос). При выборе этого варианта PowerBuilder покажет список запросов и предложит выбрать один из них. Перед этим

запрос необходимо создать в мастере запросов Query Painter, задать ему имя и сохранить в файле с расширением .pbl. Когда вы выбираете нужный запрос, который есть не что иное, как SQL-оператор SELECT с именем, PowerBuilder копирует его и использует в качестве источника данных окна DataWindow. Находясь в мастере окон данных, вы можете просматривать и изменять этот оператор с помощью Select Painter. Следовательно, как и в двух предыдущих случаях, конечным результатом является оператор SQL SELECT. PowerBuilder не организует связь окна данных с запросом, а создает копию запроса SELECT. Следовательно, если вы измените запрос, сделанные им изменения не будут отображены в окне DataWindow, и наоборот.

4. External (Внешний источник данных). Вместо того, чтобы указать PowerBuilder, как выбирать данные, вам придется самостоятельно написать код обработки данных и заполнения окна DataWindow.
5. Stored Procedure (Хранимая процедура). При выборе этого варианта источника данных DataWindow получает строки, вызывая хранимую в базе данных процедуру.

Выберем стиль представления Grid и источник данных Quick Select. Это простейшие из опций. Далее откроется диалоговое окно Quick Select. В данном диалоговом окне нужно выделить имя таблицы и имена нескольких ее столбцов.

После того, как вы выполните все операции в диалоговых окнах, PowerBuilder автоматически сформирует SQL-оператор SELECT в соответствии с вашими установками. В таблице, в нижней части диалогового окна можно определить порядок сортировки строк в окне DataWindow и критерии для отбора строк.

Создание контроллера DataWindow практически ничем не отличается от размещения в окне других элементов управления.

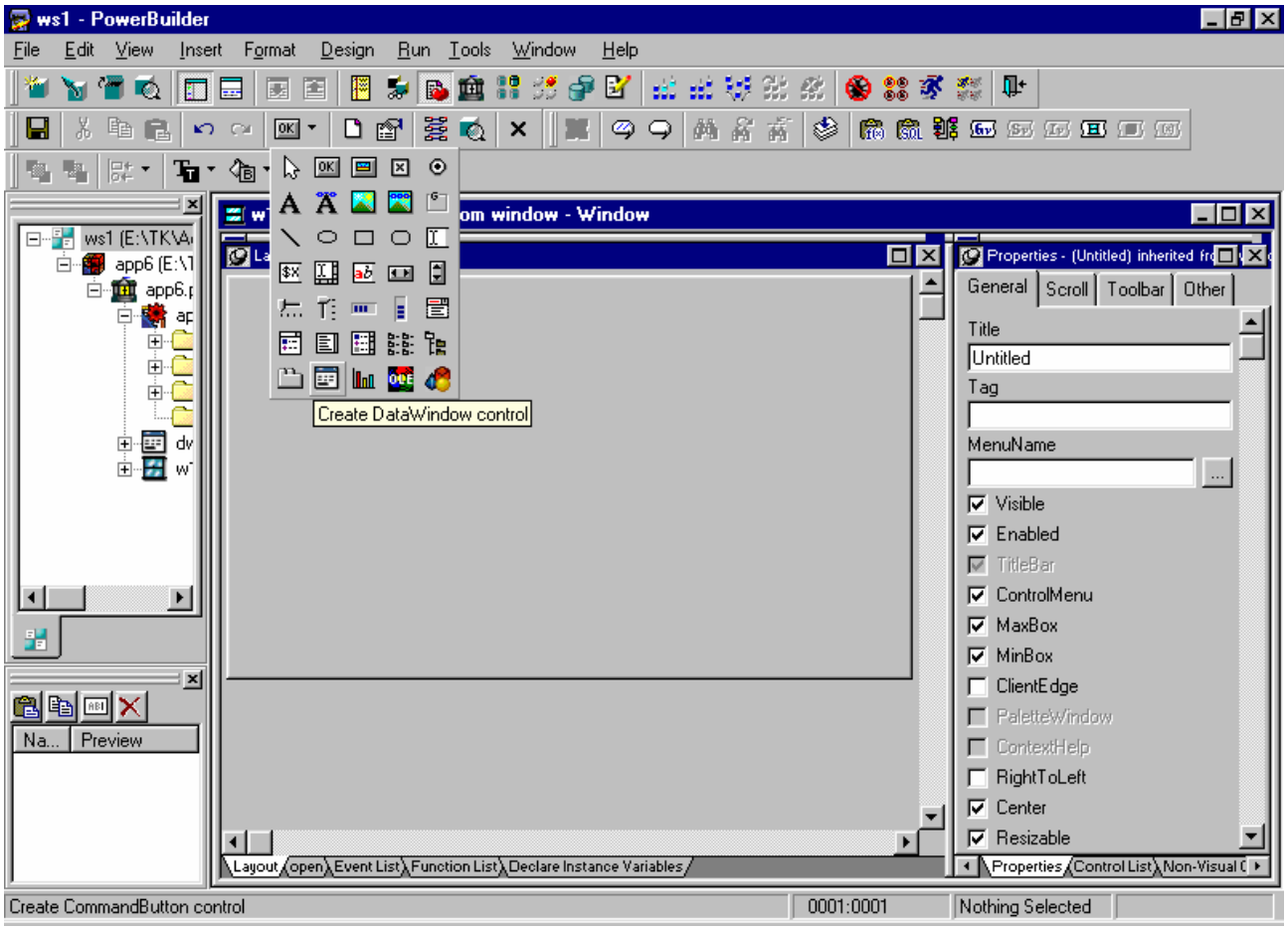

Рис. 3.16. Создание контроллера окна данных.

Для назначения объекта DataWindow контроллеру DataWindow требуется на странице свойств данного элемента управления выбрать и назначить объект DataWindow.

Обратим внимание, что объект окна данных DataWindow и контроллер DataWindow имеют разные имена.

Событию открытия окна следует назначить сценарий, который будет автоматически инициализировать DataWindow в процессе работы. (В качестве альтернативы можно использовать событие конструктор (constructor) элемента управления DataWindow).

```
dw_1.SetTransObject(sqlca)
dw_1.Retrieve()
```

Функция SetTransObject ассоциирует переменную sqlca с элементом управления DataWindow. Поэтому в любое время при работе приложения, когда через окна DataWindow на сервер будут посылаться запросы SQL,

например запрос SELECT для поиска строк или запрос UPDATE для изменения строк, то команды будут пересылаться в базу данных, к которой подключен объект транзакций sqlca.

Функция Retrieve возвращает в DataWindow строки из таблицы. Если вы не выполните эту функцию при открытии окна, то окно данных DataWindow останется пустым.

При открытии окна, содержащего DataWindow во время работы приложения происходит ассоциация DataWindow с объектом транзакции sqlca и в него возвращаются значения из таблицы.

В окне данных можно производить изменения строк таблицы, но они не будут приняты базой данных, так как не используется команда Update. Для того, чтобы поддерживать средства редактирования в окне данных DataWindow, требуется написать соответствующие программы в PowerScript.

```
dw_1.Update ()
```

Эта функция будет обновлять в базе данных те строки, в которые пользователь внесет изменения, работая в окне DataWindow.

```
dw_1.Retrieve()
```

Эта команда уже использовалась в сценарии открытия окна. Она возвращает в DataWindow значения из строк таблицы. Однако ее можно назначить отдельной кнопке окна для следующих целей:

1. Для обновления данных окна DataWindow последними данными, введенными другими пользователями базы данных.
2. Для отмены всех изменений, которые были сделаны в строках таблицы в DataWindow, без записи в базу данных и последующего обновления содержимого таблицы.

1.1.1. Работа с базой данных через окна данных

Если необходимо изменять данные БД через окно данных, то оно должно быть “updateble”. Это устанавливаем в свойствах Update Properties… (рис. 3.17.)

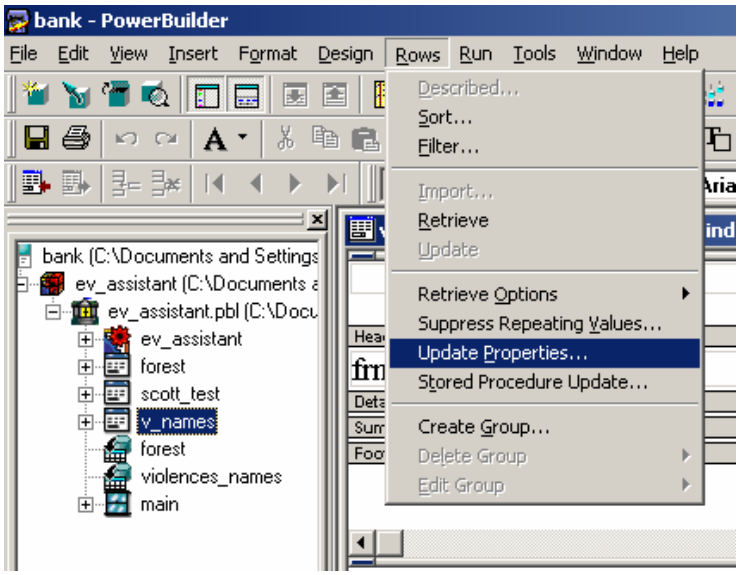

Рис. 3.17. Переход из главного меню на задание свойств режима обновления окна данных

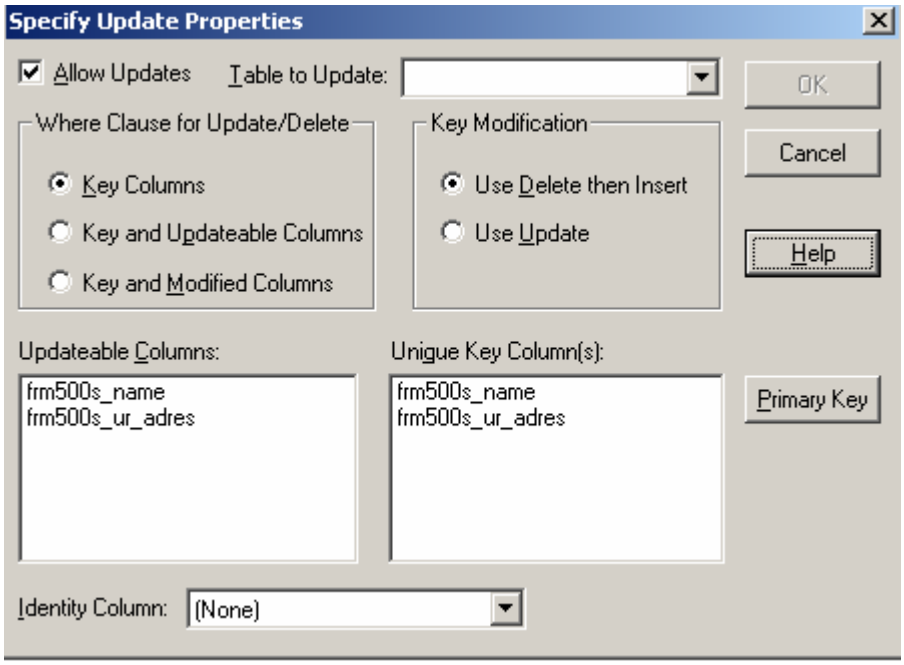

Рис. 3.18. Окно задания свойств режима обновления окна данных

Описание элементов.

Allow Updates - элемент выбора.
Необходимо выбрать, если предполагается обновление данных в таблице через объект DataWindow. По умолчанию такие обновления разрешены, если окно данных содержит поля только одной таблицы и в них входят ключевые поля.

Table to Update - список
Если окно данных содержит поля нескольких таблиц, то необходимо указать имя таблицы, в которой будут обновления через данное окно данных.

Where Clause for Update/Delete - группа радио кнопок.

Определяет, какие поля включаются в опцию **WHERE** операторов **Update/Delete**.

Замечание: Выбирайте здесь вариант **Key Columns** только, если БД используется в монопольном режиме одним пользователем или применена блокировка БД.

Key Modification - группа радио кнопок.
Определяет, какие генерируются операторы SQL, когда поле, указанное в окне **Unique Key Column(s)** изменено.

Use DELETE then INSERT используется, если в СУБД не разрешено пользователям модифицировать два ключа или устанавливать какое – либо значение поля в одной записи, равное значению поля в другой записи, или если СУБД имеет триггер Insert.

Use UPDATE используется, если в СУБД только одна запись может быть изменена перед обновлением БД.

Updateable Columns - окно
В окне можно выбрать и подсветить обновляемые поля. Если такие поля выбраны, убедитесь, что их порядковые номера табуляции отличны от нуля – тогда их может выбрать пользователь при работе приложения.

Unique Key Column(s) - окно
Здесь указываются поля, которые идентифицируют обновляемую запись. По умолчанию считаются поля первичного ключа.

Primary Key- кнопка.
Выбирается, чтобы вместо окна Unique Key Column(s) использовать первичные ключи.

Identity Column

Указывается особое поле, величина в котором в новой записи устанавливается СУБД. Если такое поле установлено, то PowerBuilder отображает его величину в новой вставляемой в окно данных строке.

Выводы:

- если окно данных содержит поля **только одной таблицы** и среди них есть ключевое поле, то они автоматически задаются как обновляемые поля с ненулевыми значениями номеров, определяющих порядок выбора таких полей; пользователи могут менять данные в таких полях непосредственно в окне данных.

- если окно данных содержит поля **нескольких таблиц или вида,** то они автоматически задаются как **необновляемые поля** с нулевыми значениями номеров, определяющих порядок выбора таких полей; пользователи не могут менять данные в таких полях в окне данных;

- пользуясь окном на рис. 2, можно разрешать или запрещать обновления некоторых полей в окне данных, **но только принадлежащих одной таблице**!

Пример. Имеем таблицы студентов и их адресов:

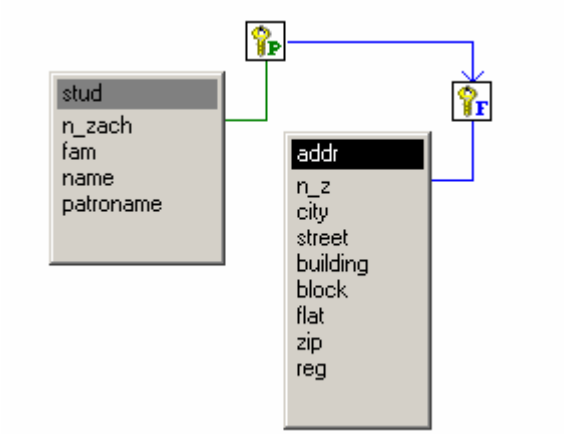

Рис. 3.19. Пример структуры связанных таблиц.

Создадим объект - окно данных типа Quick Select.

Н рис. 3.20 показаны структура его источника данных и оператор запроса.

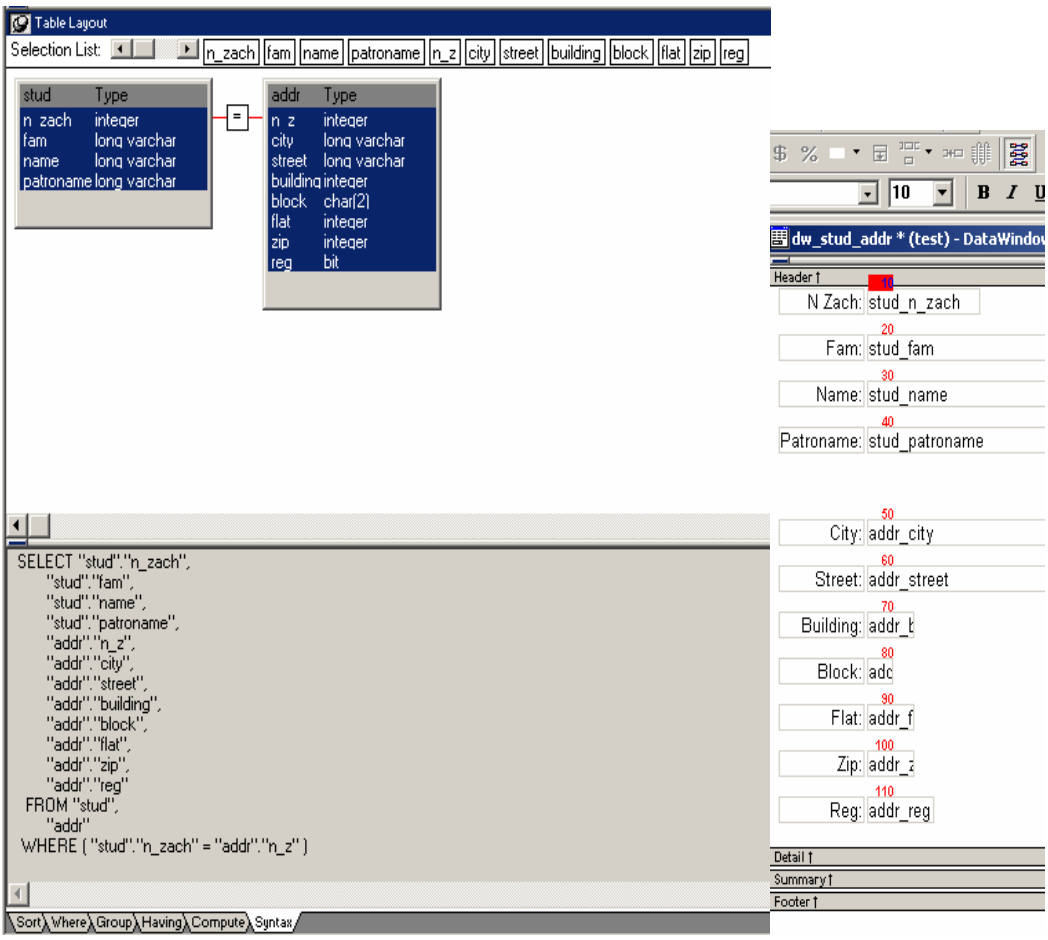

Рис. 3.20. Макет источника данных.

Зададим условия обновления для одной из таблиц: stud

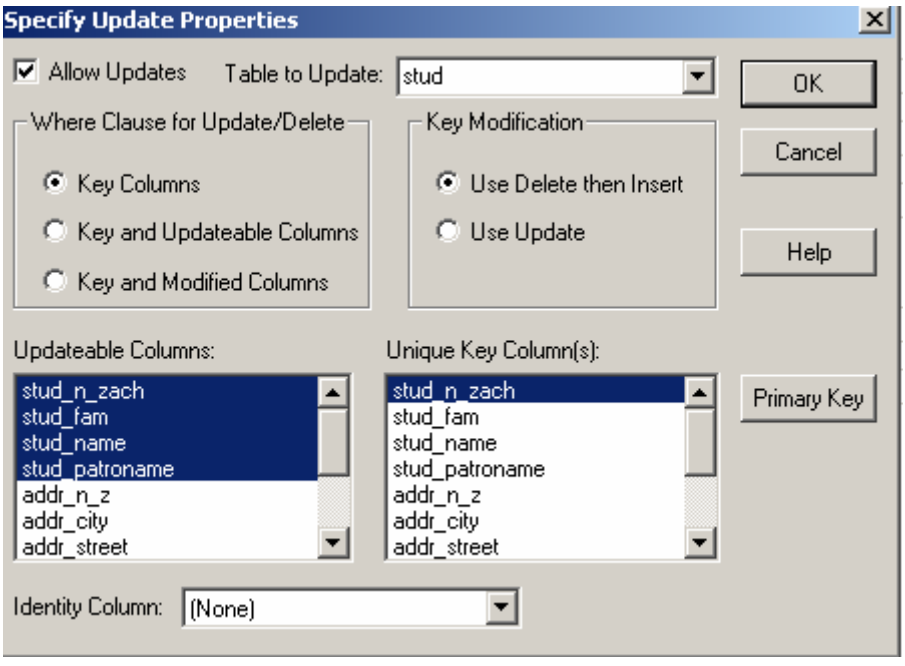

Рис. 3.21. Порядок выбора полей таблиц

Назначим номера, определяющие порядок выбора всех полей всех таблиц в окне данных.

Нажмем клавишу  системного меню и расставим номера у всех полей

После этого станет возможным редактирование всех полей в окне данных. Но по скрипту

dw_1.Update() обновятся только поля таблицы stud

Поэтому скрипт должен быть другой:

```
long cr
cr=dw_1.GEtROw()
boolean lb_ok
string is_updateerror
integer li_return

lb_ok = True      // пока не обнаружено ни одной ошибки
is_updateerror = ""  //пока нет сообщений об ошибке

li_return = dw_1.Update(True,False) //Update в таблице stud

IF li_return <> 1 OR is_updateerror <> "" THEN
lb_ok = False
END IF
IF lb_ok THEN
  dw_1.object.DataWindow.Table.UpdateTable = "addr"
dw_1.object.addr_city.Update = "yes"
dw_1.object.addr_street.Update = "yes"
dw_1.object.addr_building.Update = "yes"
dw_1.object.addr_block.Update = "yes"
dw_1.object.addr_flat.Update = "yes"
dw_1.object.addr_zip.Update = "yes"
dw_1.object.addr_reg.Update = "yes"
dw_1.object.addr_n_z.Update = "yes"

dw_1.object.stud_n_zach.Key = "no"
dw_1.object.stud_n_zach.Update = "no"
dw_1.object.stud_fam.Update = "no"
dw_1.object.stud_name.Update   = "no"
dw_1.object.stud_patroname.Update   = "no"

dw_1.object.addr_n_z[cr] = dw_1.object.stud_n_zach[cr] // здесь - номер строки
аргумент

li_return = dw_1.Update(True,False) //Update в таблице addr

IF li_return <> 1 OR is_updateerror <> "" THEN
lb_ok = False
```

```
END IF

// Возвращаем в исходное
dw_1.object.DataWindow.Table.UpdateTable = "stud"
dw_1.object.addr_city.Update = "no"
dw_1.object.addr_street.Update = "no"
dw_1.object.addr_building.Update = "no"
dw_1.object.addr_block.Update = "no"
dw_1.object.addr_flat.Update = "no"
dw_1.object.addr_zip.Update = "no"
dw_1.object.addr_reg.Update = "no"
dw_1.object.addr_n_z.Update     = "no"
dw_1.object.stud_n_zach.Key  = "yes"
dw_1.object.stud_n_zach.Update = "yes"
dw_1.object.stud_fam.Update = "yes"
dw_1.object.stud_name.Update   = "yes"
dw_1.object.stud_patroname.Update   = "yes"

END IF

IF lb_ok THEN
COMMIT;
dw_1.ResetUpdate()
ELSE
ROLLBACK;

MessageBox("Error - Обновление БД не прошло", &
is_updateerror, StopSign!)
END IF
```

Пример 2. Другой вариант. Обновление двух объектов DataWindow

```
int rtncode

CONNECT USING SQLCA;

dw_cust.SetTransObject(SQLCA)

dw_sales.SetTransObject(SQLCA)

rtncode = dw_cust.Update(TRUE, FALSE)

IF rtncode = 1 THEN
        rtncode = dw_sales.Update(TRUE, FALSE)
        IF rtncode = 1 THEN
                dw_cust.ResetUpdate()    // Both updates are OK
                dw_sales.ResetUpdate()   // Clear update flags
                COMMIT USING SQLCA;   // Commit them
        ELSE
                ROLLBACK USING SQLCA; // 2nd update failed
        END IF
END IF
```

1.1.2. Как работают окна данных

Когда приложение выбирает строки из базы данных, в окно Data Window, система сохраняет эти строки в *буферах*. Обычно буферы находятся в ОЗУ компьютера клиента. Буферы содержат не только отображаемые на экране строки, но и все остальные данные, выбранные запросом SELECT окна Data Window.

Существует четыре буфера:

- Первичный (Primary) буфер
- Буфер фильтра (Filter)
- Буфер удаления(Delete)
- Буфер оригинала (Original)

В первичном буфере находятся строки, непосредственно находящиеся в окне данных.

Буфер фильтра содержит строки, выбранные с помощью команды SELECT, но затем отфильтрованные дополнительными условиями.

Буфер удаления содержит строки, которые сначала были в первичном буфере, но затем удалены с помощью функции DeleteRow.

Буфер оригинала содержит строки, которые были выбраны из БД функцией Retrieve(), до того, как пользователь начал изменять их.

1.1.2.1. Доступ к буферам

Power Builder позволяет работать непосредственно с буферами окна данных.

Значения каждой строки и каждого столбца первичного буфера образуют двумерный массив

data[r,c],

где r - номер строки, c - номер столбца. Обращаться к буферам можно непосредственно, указывая их имена в выражениях, например:

```
v=dw_1. object. a.filter[r]
v=dw_1. object. a.delete[r]
v=dw_1. object. a.delete.original[r,c]
```

Еще примеры доступа

В окне данных dw_employee строковой переменной LName присваивается значение из поля emp_name строки 3 в первичном буфере окна данных. Здесь номер буфера не указан явно, поэтому принимается его значение по умолчанию, соответствующее первичному буферу.

```
String LName

LName = dw_employee.GetItemString(3, "emp_name")
```

Строковой переменной **LName** присваивается значение из поля emp_name строки 3 в буфере удаления окна данных.

```
String LName

LName = dw_employee.GetItemString(3, &

    "emp_name", Delete!)
```

Этот пример отличается от предыдущего тем, что для данных в буфере удаления устанавливается признак TRUE, означающий, что обрабатываются данные, первоначально помещенные в этот буфер из базы данных.

```
String LName

LName = dw_employee.GetItemString(3, &
    "emp_name", Delete!, TRUE)
```

1.1.2.2. Флажки состояния буферов

Для каждого поля каждой строки каждого буфера (кроме буфера оригинала) в PowerBuilder существует флажок состояния (статус). Он может принимать одно из двух значений:

• **notmodified!**— поле не было изменено; если это новая строка, то значение поля еще не установлено, в противном случае — поле содержит значение, считанное из базы данных;

- **datamodified!** — поле было изменено.

PowerBuilder управляет соответствующими флажками состояний и для каждой строки в каждом из трех флажок состояния строки в PowerBuilder может иметь одно из четырех значений;

- **notmodified!** — это не новая строка (т.е. строка была получена из базы данных) и она не была изменена (т.е. этой строки имеют статус notmodified!);
- **datamodified!** — это не новая строка (т.е. строка была получена из базы данных) и она уже была изменена (т.е. по крайней мере, одно ее поле имеет статус datamodified!).
- **new!** — это новая строка (т.е. строка вставлена, а не получена из базы данных) и она еще не заполнена (т.е. все еще имеют статус notmodified!).

- **newmodified!** — это новая строка (т.е. строка вставлена, а не получена из базы данных) и уже частично заполнена (т.е. хотя бы одно из полей имеет статус datamodified!).

Флажки состояния играют важную роль при формировании запросов на обновление базы данных, таких как UPDATE и DELETE. Флажки часто используются в сценариях, помогая определить, что было изменено, а что нет

Флажки состояния не нужны для полей и строк буфера оригинала, поскольку значения в этом буфере не изменяются.

А для полей и строк буфера удаления флажки нужны. Хотя строки и поля в этом буфере могут быть удалены, их можно восстановить в первичный буфер. В этом случае PowerBuilder (и, возможно, ваши сценарии) должны знать были ли они изменены.

1.1.2.3. Изменение значений флажков состояния в PowerBuilder

Для чтения и изменения значений флажков состояния используются две функции: GetItemStatus и SetItemStatus. Следующий оператор проверяет, был ли изменен столбец *c* строки *r* в буфере фильтра:

```
IF dw_l.GetItemStatus(r,c,filter!) = datamodified! THEN ...
```

Изменить статус того же элемента, чтобы считалось, что элемент был изменен, можно с помощью следующего оператора:

```
dw_l.SetItemStatus(r, c, filter!, datamodified!)
```

Обе эти функции работают как с флажками состояния строки, так и с флажками состояния полей. Для управления флажком состояния строки в качестве номера столбца используется нуль:

```
IF dw_l.GetItemStatus(r, 0, filter!) = newmodified! THEN ...
dw_l.SetItemStatus(r, 0, filter!, newmodified!)
```

Пользоваться функцией SetItemStatus следует очень аккуратно, поскольку флажки состояния играют очень важную роль в процессе обновления данных окна DataWindow.

1.1.2.4. Жизненный цикл буферов DataWindow.

Поняв назначение каждого буфера DataWindow, легко понять, как влияет на них каждая стандартная операция в DataWindow.

- **Функция Retrieve** очищает буферы; помещает все возвращенные строки в буфер оригинала; в зависимости от используемого фильтра помещает

каждую возвращенную строку либо в первичный буфер, либо в буфер фильтра; устанавливает флажок состояния каждой строки и каждого столбца в состояние notmodified!.

- **Функция InsertRow** вставляет новую строку в первичный буфер; флажку состояния этой строки присваивает значение new!, а флажкам состояния каждого столбца (поля) этой строки —значение notmodified!.
- **Функция DeleteRow** выполняет следующие действия: если флажок состояния строки установлен в notmodified! или datamodified!, функция DeleteRow перемещает эту строку из первичного буфера в буфер удаления, не изменяя значение флажков состояния. Если флажок состояния строки установлен в положение new! или newmodifled!, то строке удаляется из первичного буфера, без передачи в буфер удаления.
- **Изменение значения поля** устанавливает флажок состояния поля в datamodified!, изменяя флажок состояния строка на datamodified, если он был установлен в notmodified!, или на newmodifled, если его значение — new!.
- **Функция Update** генерирует SQL-запрос DELETE для каждой строки в буфере удаления; генерирует SQL- запрос (INSERT для каждой строки с флажком состояния newmodifled! в буфере фильтра или в первичном буфере; генерируем SQL-запрос UPDATE для каждой строки с флажком состояния datamodified! в буфере фильтра или в первичном буфере. В запросе UPDATE в предложение SET включаются только поля с флажком состояния datamodified!

После выполнения SQL-запросов функция Update очищает буфер удаления, сбрасывает флажки всех строк и столбце в состояние notmodified! (за исключением строк с флажком состояния new!, который остается прежним, т. е. new!) и обновляет буфер оригинала текущими значениями первичного буфера и буфера фильтра.

Функция Update имеет необязательный параметр, с помощью которого можно запретить сброс содержимого буферов и флажков состояния. Другая функция ResetUpdate сбрасывает содержимое буферов и флажки состояния без генерации SQL-запросов. Эти возможности могут быть полезны, если в приложении необходимо более сложное управление процессом обновления данных, чем обычно. Например, если требуется синхронно обновлять данные в двух окнах DataWindows.

На самом деле для ускорения процесса считывания работа буфера оригинала PowerBuilder организована более сложно, чем здесь описано. Функция Retrieve не заполняет буфер оригинала строками данных — буфер остается пустым. Лишь при необходимости PowerBuilder копирует строки в буфер оригинала, перед тем, как пользователь или приложение начнут

модифицировать данные. Тем не менее разработчику вполне достаточно представлять процесс так, как он описан.

1.1.2.5. Манипулирование буферами в PowerScript

В предыдущем разделе описаны многие функции, предназначенные для работы с буферами DataWindow, например InsertRow, DeleteRow и GetItemStatus. Ниже перечислены следующие функции:

- **RowCount** возвращает число строк в первичном буфере.
- **FilteredCount** возвращает число строк в буфере фильтра.
- **DeletedCount** возвращает число строк в буфере удаления.
- **ModifiedCount** возвращает число строк с флажками datamodified! и newmodified!, содержащимися в первичном буфере и в буфере фильтра.
- **RowsMove** перемещает одну или более строк из одного буфера (первичного буфера, буфера фильтра или буфера удаления) в другой буфер того же окна DataWindow или другого окна.
- **RowsCopy** работает так же, как RowsMove, но вместо того, чтобы перемещать строки, копирует их.
- **RowsDiscard** удаляет одну или несколько строк из буфера (первичного буфера, буфера фильтра или буфера удаления но в отличие от функции DeleteRow, не перемещает их в буфер удаления.
- **Reset** очищает окно DataWindow, стирая все буферы.

Суть технологий, элементами которых являются механизмы управления буферами, проявляется в таких задачах обработки данных, где запросы выполняются на нескольких таблицах одновременно.