МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего профессионального образования

«Тульский государственный университет»

Кафедра «Автоматика и телемеханика»

Методические указания

ПО ВЫПОЛНЕНИЮ КУРСОВОГО ПРОЕКТА (РАБОТЫ)

по дисциплине

Базы данных

Направления подготовки**:**

230100 Информатика и вычислительная техника
230400 Информационные системы и технологии
230700 Прикладная информатика
231000 Программная инженерия

Квалификация (степень) выпускника: *бакалавр*

Формы обучения: *очная*

Тула 2012 г.

Методические указания курсового проекта (работы) составлены *профессором, д.т.н. Богатыревым М.Ю.* и обсуждены на заседании кафедры *автоматики и телемеханики* факультета *кибернетики*,

протокол № 6 от " 31 " января 2012 г.

Зав. кафедрой_____А.А. Фомичев

Методические указания курсового проекта (работы) пересмотрены и утверждены на заседании кафедры *автоматики и телемеханики* факультета *кибернетики*,

протокол №___ от "___"_____ 20___ г.

Зав. кафедрой_____А.А. Фомичев

1. ВВЕДЕНИЕ. ЦЕЛИ, ЗАДАЧИ И ЭТАПЫ КУРСОВОГО ПРОЕКТИРОВАНИЯ.

Курсовое проектирование по дисциплине "*Базы данных*" выполняется с целью приобретения практических навыков разработки и исследования информационных систем, содержащих базы данных.

Задачами курсового проектирования являются:

- проектирование базы данных, соответствующей выбранной модели данных;
- выбор архитектуры системы управления базой данных (СУБД);
- разработка программного обеспечения СУБД для выбраннойархитектуры;
- проверка работоспособности созданной СУБД.

Проектирование современных информационных систем является сложным многозвенным процессом. Он опирается на понятие *жизненного цикла информационной системы* (ЖЦ)[1-9].

На рис. 1 показана структура модели жизненного цикла информационной системы. Поддержка жизненного цикла обеспечивается итерационным процессом проектирования.

Рассмотрим этапы проектирования.

На этапе *стратегического планирования и анализа* проводится маркетинговое исследование потребностей потенциального пользователя или анализ требований заказчика, исследование прикладной области, в которой будет использоваться система, проверка технической реализуемости, экономическое обоснование, формирования исходного плана по распределению различного рода ресурсов (людских, материальных, технических и т.д.), выбираются возможные методологии и средства создания системы, определение сроков реализации для каждого этапа ЖЦ и порядка контроля качества, а также инициализация проекта. Проще говоря, на этом этапе происходит четкая постановка задачи и предварительный анализ путей и средств решения. Результатом в конце этого этапа являются спецификации требований и входных данных, бизнес-план организации деятельности по созданию информационной системы.

На этапе *проектирования* определяются первичные данные, формируется структура данных, проводится анализ существующих систем и создается проект архитектуры в соответствии с выбранной методологией разработки системы и средствами разработки. Результатом на этапе проектирования должны быть проекты моделей базы данных, моделей функций, модели интерфейсов, модели архитектуры и требования предъявляемые к ним, различная документация (спецификация требований к системе, требования по созданию и т.д.), а также уточняется план создания системы.

На этапе *разработки* создаются модули интерфейсов, экранов, отчетов, пакетных процессов и текстов помощи (HELP- файлы), появляется база данных, описывается конфигурация версии системы, оформляется документация по использованию системы и, кроме этого, формируются проекты тестов и уточняется план интеграции и тестирования. Результатом этого этапа является готовая система и пользовательская документация.

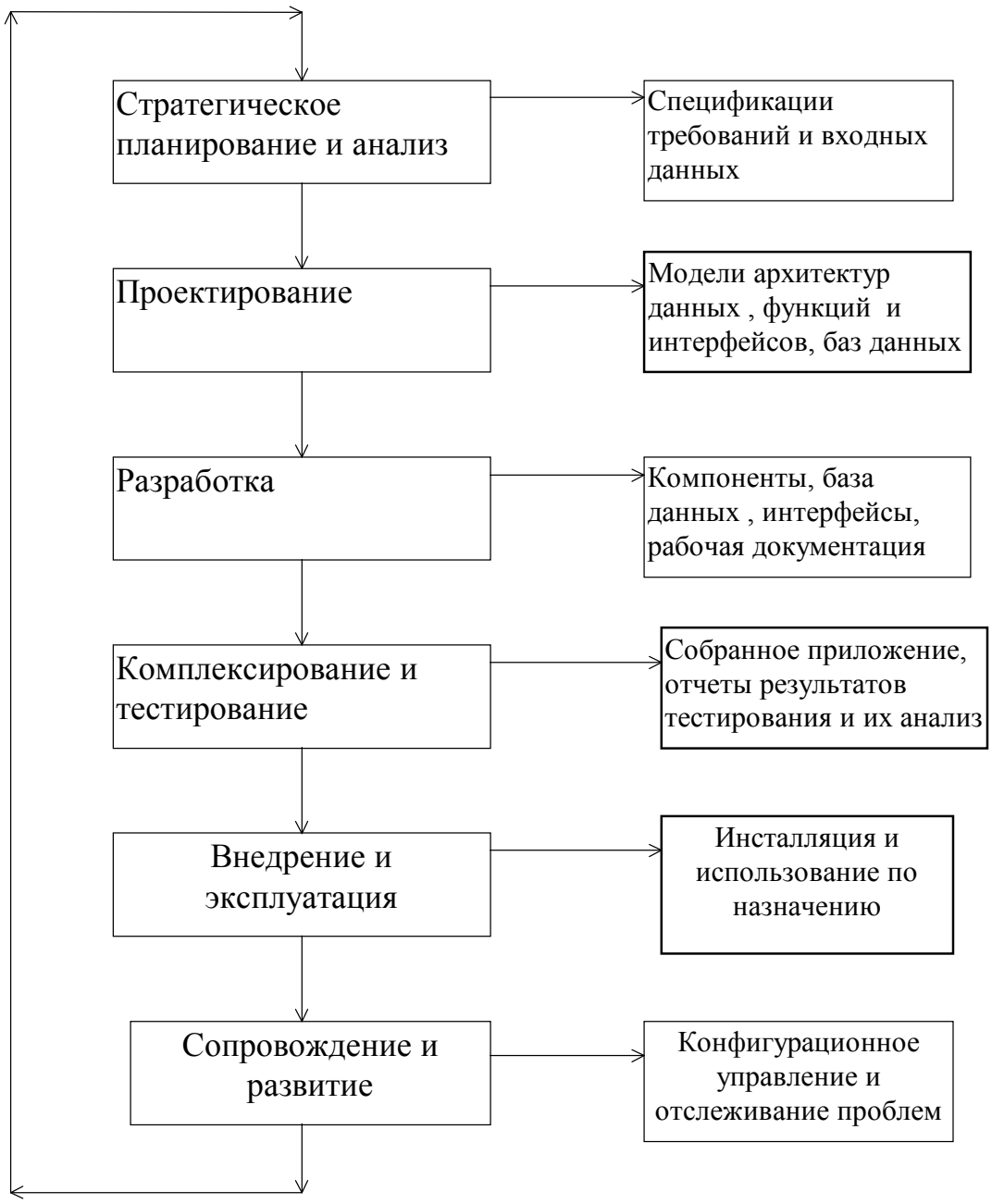


Рис. 1.1. Модель жизненного цикла информационной системы.

На этапе *интеграции и тестирования* создается варианты тестов в зависимости от вида тестирования, формируется база тестовых данных, проверяются отдельные компоненты системы, тестируется система в целом, оформляется отчет по результатам тестирования как отдельных компонент системы так и в целом. Результатом этого этапа является отчет и анализ тестирования системы, возможности для корректирования очередной ее версии.

На этапе *внедрения и эксплуатации* происходит установка и наладка клиентских мест и серверной части системы пользователя, обучается пользовательский персонал, а также в процессе эксплуатации, в классическом понимании, происходит транспортировка, хранение и поддержании работоспособности и использование по прямому назначению.

На этапе *сопровождения и развития* происходит сбор информации о качестве системы, модификация ее программного обеспечния (ПО), текущее описание конфигурации, извещение пользователя о внесенных изменениях. Стадия сопровождения обусловлена необходимостью выполнением следующих задач: включения новых функций, изменения функций, обнаружения ошибок и их исправления. Цель, в данном случае, заключается в изменении функциональных свойств существующего ПО при сохранении его целостности. К процессу сопровождения относятся также действия по переносу или удалению ПО. На этом этапа осуществляется конфигурационное управление ПО. Результатом этого этапа является база данных проблем и соответствующих модификаций ПО.

Таким образом, модель жизненного цикла представляет собой *систему с обратной связью*. Соответственно, системы, строящиеся согласно данной модели, обладают важным свойством *развиваемости*.

Курсовой проект информационной системы, является учебной разработкой и не может претендовать на полную поддержку жизненного цикла приложения. Однако, основные элементы технологии, показанной на рис.1, должны быть реализованы при курсовом проектировании.

Здесь имеют место следующие особенности.

Этап *стратегического планирования и анализа* соответствует формированию задания на КР и его анализу.

На этапе *проектирования* происходит выбор модели данных для информационной системы и форматов хранения данных в базах.

На этапе *разработки* создается приложение, работающее с базами данных - СУБД.

Этап *интеграции и тестирования,* как правило, не выделяется в отдельный раздел работы. Хотя возможны задания на разработку модулей, которые должны быть интегрированы в уже имеющуюся систему. В этом случае необходимо тестирование, по крайней мере, как гарантия того, что вновь созданный модуль не нарушит работу исходной системы.

Этапу *внедрения и эксплуатации* соответствует установка и запуск готовой информационной системы. Здесь не требуется обучение какого-либо персонала.

Хорошо, если хотя бы сам автор системы достаточно подготовлен, чтобы грамотно установить в операционной среде собственное произведение.

На этапе *сопровождения и развития* исправляются ошибки в готовой системе, уточняется ее документация и фиксируются необходимые результаты работы системы.

2. СОДЕРЖАНИЕ КУРСОВОЙ РАБОТЫ (ПРОЕКТА)

Рассмотренные выше этапы проектирования реализуются в создаваемой студентом информационной системе и должны быть отражены в пояснительной записке к курсовой работе.

В пояснительной записке должна быть представлена следующая информация.

2.1. Анализ задания

Задание на проектирование оформляется в виде стандартного бланка задания. В бланке задания должны быть помещены следующие данные:

- назначение проектируемой информационной системы, следующее чаще всего из ее названия;
- характер данных используемых системой;
- объем данных в базах;
- возможные требования к форматам данных;
- возможные требования к интерфейсу системы;
- возможные требования к инструментальной среде разработки;
- выходные документы системы.

Каждое конкретное задание может содержать только часть указанного перечня или иметь особые требования.

Анализ задания состоит в том, чтобы прежде всего определить следующее:

форматы данных информационной системы, т.е. конкретный тип базы данных, применяемой в системе;

инструментальную среду разработки системы, ее инструментальную СУБД.

Решение этих вопросов связано друг с другом. Современные инструментальные СУБД, как правило, поддерживают несколько форматов баз данных. Поэтому можно, казалось бы, не обращать внимание на конкретный формат. Тем не менее, выбор типа базы данных важен. Вопросы выбора современных БД освещены в [10] . Принципиально важно то, к какому типу будет отнесена проектируемая БД - к *локальному* или *сетевому* (распределенному).

2.2. Проектирование базы данных

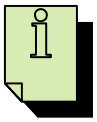
Для проектирования баз данных применяется концептуальное, логическое и физическое моделирование. Современные CASE – системы, такие как Sybase PowerDesigner, позволяют выполнить все этапы моделирования и далее реализовать базу данных в выбранной платформе СУБД. Применение CASE – систем в курсовом проектировании предпочтительно, поскольку позволяет студентам овладеть современными технологиями проектирования информационных систем.

Концептуальная модель «сущность - связь» создается на первом этапе моделирования баз данных. Модель основана на понятиях «сущность» и «связь», с которыми студенты знакомятся на лекциях и по литературе [1, 2, 3, 5].

При применении системы Sybase PowerDesigner для создания модели «сущность - связь» и последующей ее обработки необходимо учитывать следующие особенности.

Нотация моделирования. Нотация — это система условных обозначений, принятая в какой-либо области знаний или деятельности. Система PowerDesigner поддерживает все современные нотации, используемые в концептуальном моделировании. Несмотря на то, что нотация — это всего лишь внешняя сторона модели, разные нотации в системе PowerDesigner обеспечивают различные возможности моделирования.

По умолчанию в системе используется *нотация Баркера*.



Заметьте, что в нотации Баркера в панели инструментов в мастерской концептуального моделирования не активны три инструмента: *наследование*, *ассоциация* и *ассоциативная связь* — см. рис. 2.1.

Выбрав другую нотацию, например, E/R+Merise, получим полный набор инструментов концептуального моделирования.

2.2.1. Применение расширенной модели «сущность – связь»

Нотации, использующие полный набор инструментов концептуального моделирования, позволяют применять *расширенную модель «сущность – связь»*, реализуемую посредством диаграмм EERD.

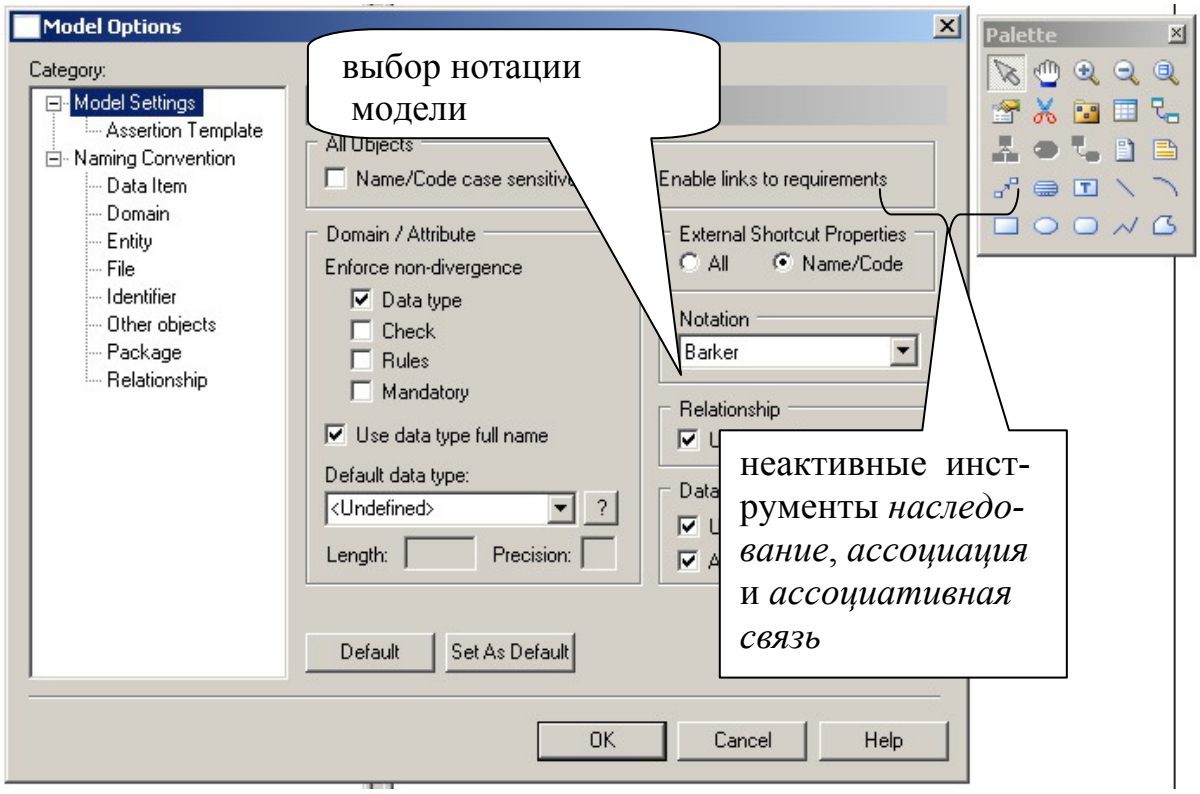


Рис. 2.1. Инструменты концептуального моделирования в нотации Баркера.

Рассмотрим особенности применения диаграмм EERD.

2.2.1.1. Применение иерархий сущностей.

Сущности, используемые в задании на курсовую работу, могут быть связаны иерархично. Например, в СУБД «ВУЗ» необходимо хранить данные о студентах и сотрудниках. К последним относятся преподаватели, лаборанты, инженеры, проректоры и т.д. Все они – люди, поэтому наследуют атрибуты «фамилия», «имя», «отчество», «дата рождения», «пол» сущности «*человек*». На рис. 2.2. показана диаграмма EERD, в которой применено наследование сущностей.

Обратите внимание на ключи сущностей и на то, что наследуемые атрибуты не включаются в сущности - наследники.

В окне свойств элемента модели «Наследование» (Inheritance) в закладке «Генерация» (Generation) необходимо выбрать генерацию таблиц как для родительской сущности, так и для дочерних – см. рис. 2.2.

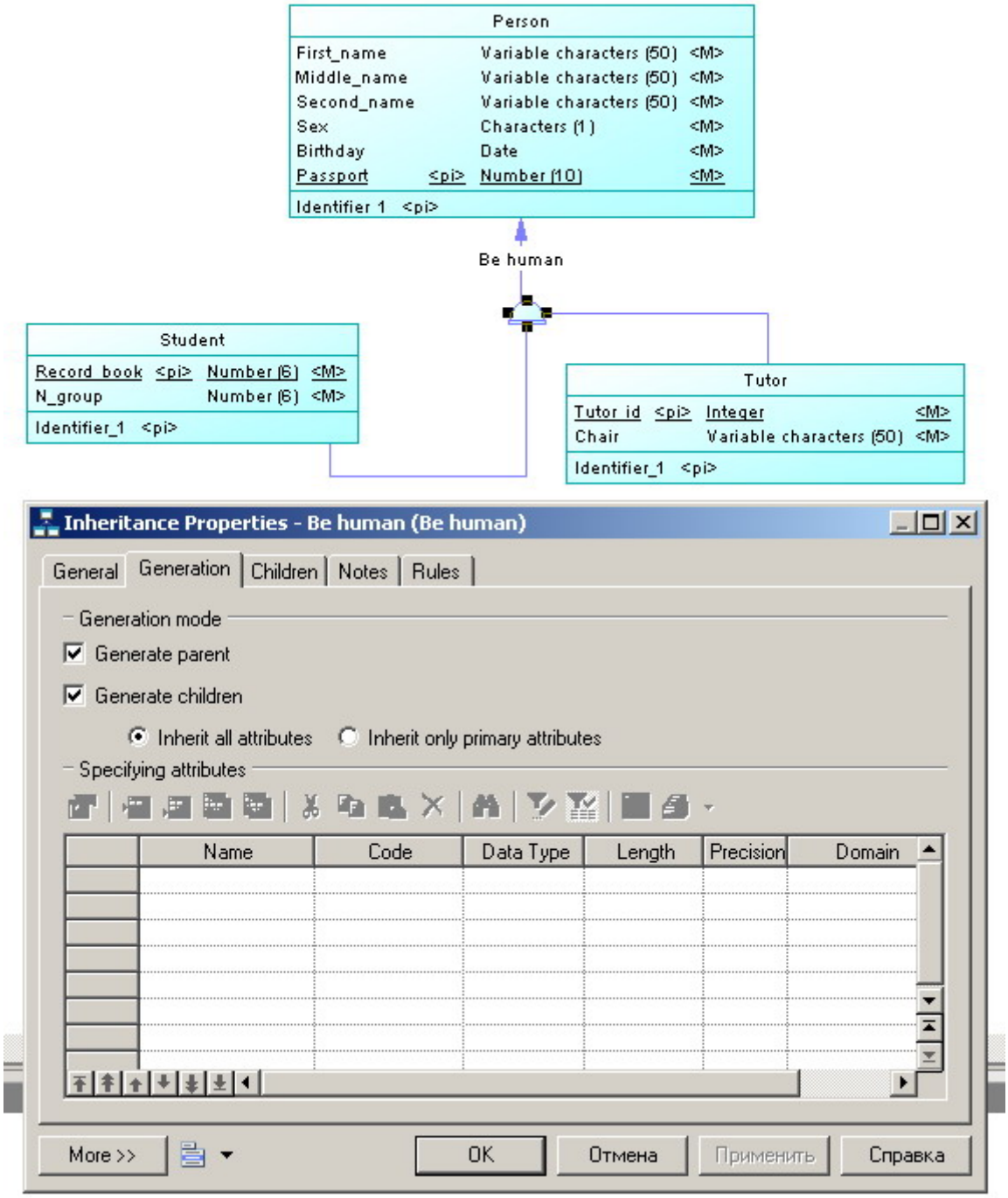


Рис. 2.2. Диаграмма EERD, в которой применено наследование сущностей, и окно свойств элемента «Наследование» (Inheritance).

При построении физической модели для модели, использующей наследование, имеется два варианта генерации:

- с наследованием всех атрибутов родительской сущности и
- с наследованием только ключевых атрибутов родительской сущности.

Выбрав второй вариант, получаем физическую модель, показанную на рис. 2.3.

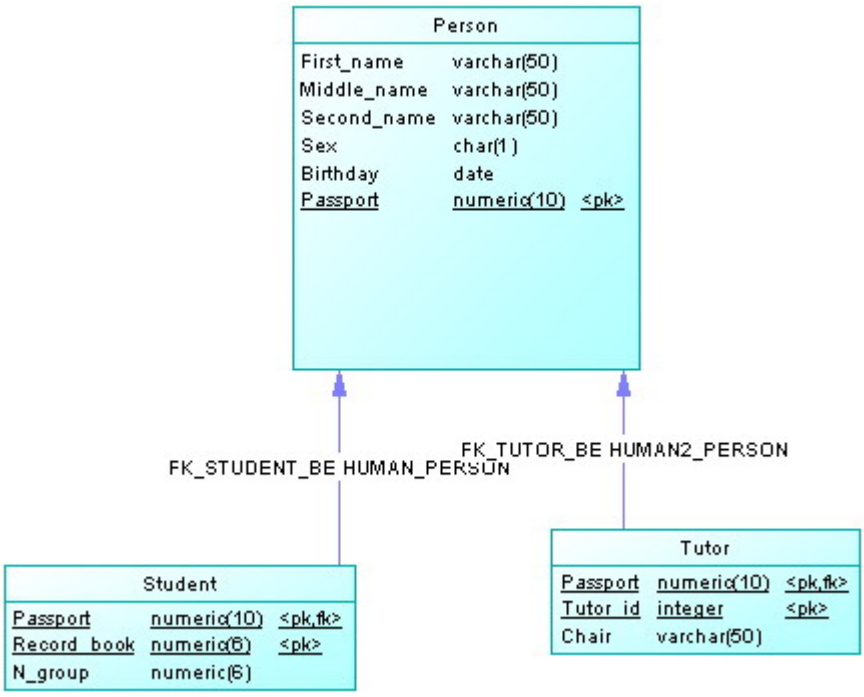
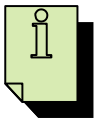


Рис. 2.3. Физическая модель для диаграммы EERD на рис. 2.2.

На основании результата построения физической модели базы данных по диаграмме EERD можно сформулировать общее правило.



Наследование сущностей, имеющееся в диаграмме EERD, реализуется в физической модели посредством помещения ключа родительской сущности в качестве внешних ключей в таблицы, соответствующие сущностям – наследникам.

Особенность данного правила в том, что оно действует на любом числе сущностей – наследников. Как известно, правила, применяемые для построения физических моделей по ERD (правила Джексона), применимы только к парам связанных сущностей.

Синхронизация данных. Принцип наследования требует синхронизации изменений данных в родительских и дочерних объектах. В физической модели на рис. 2.3. это имеет место на ключах: изменение значений первичного ключа родительской таблицы отражаются в изменениях значений внешних ключей дочерних таблиц.

В системе PowerDesigner можно наследовать все атрибуты родительской сущности, выбрав опцию Inherit all attributes, как показано на рис. 2.2. В этом случае синхронизации подлежат не только ключевые атрибуты. Целесообразность такого решения определяется в каждом конкретном случае. В примере на рис. 2.2. – 2.3.наслдовать все атрибуты нецелесообразно.

2.2.1.2. Применение ассоциаций.

На практике несколько сущностей могут быть связаны или *ассоциированы* посредством определенных процессов. Такая связь называется *ассоциацией*.

Ассоциация не является сущностью, но обладает атрибутами. Среди атрибутов ассоциации нет ключевых.

На рис. 2.2. показано развитие рассматриваемой модели путем добавления связи сущностей «студент», «преподаватель» и «дисциплина» посредством ассоциации «экзамен».

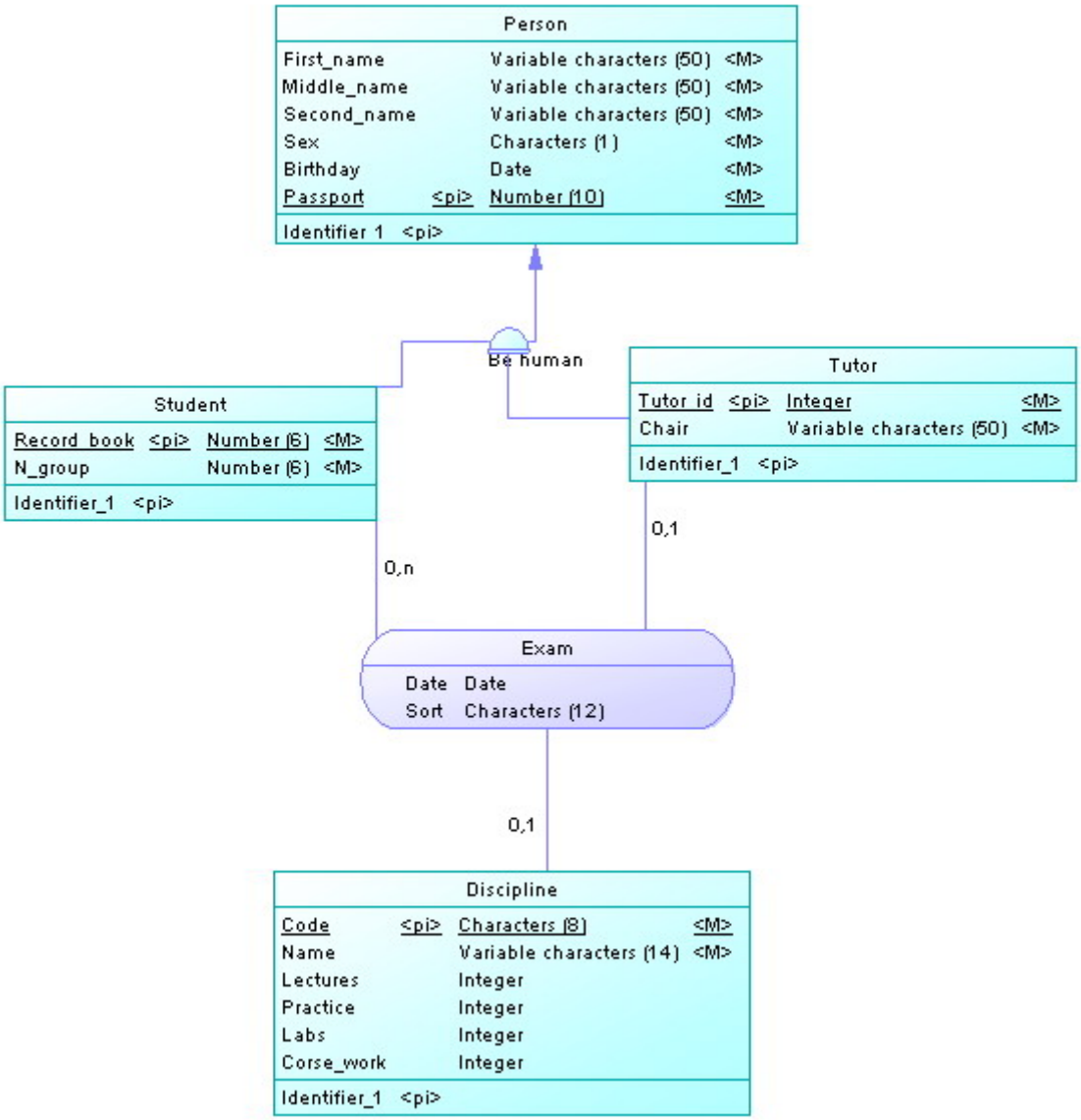


Рис. 2.2. Применение ассоциации для связывания сущностей.

Ассоциация экзамен имеет два атрибута: дата экзамена и тип экзамена – письменный или устный.

Связи, установленные между ассоциацией и сущностями, определяют обмен атрибутами в таблицах физической модели, показанной на рис. 2.5.

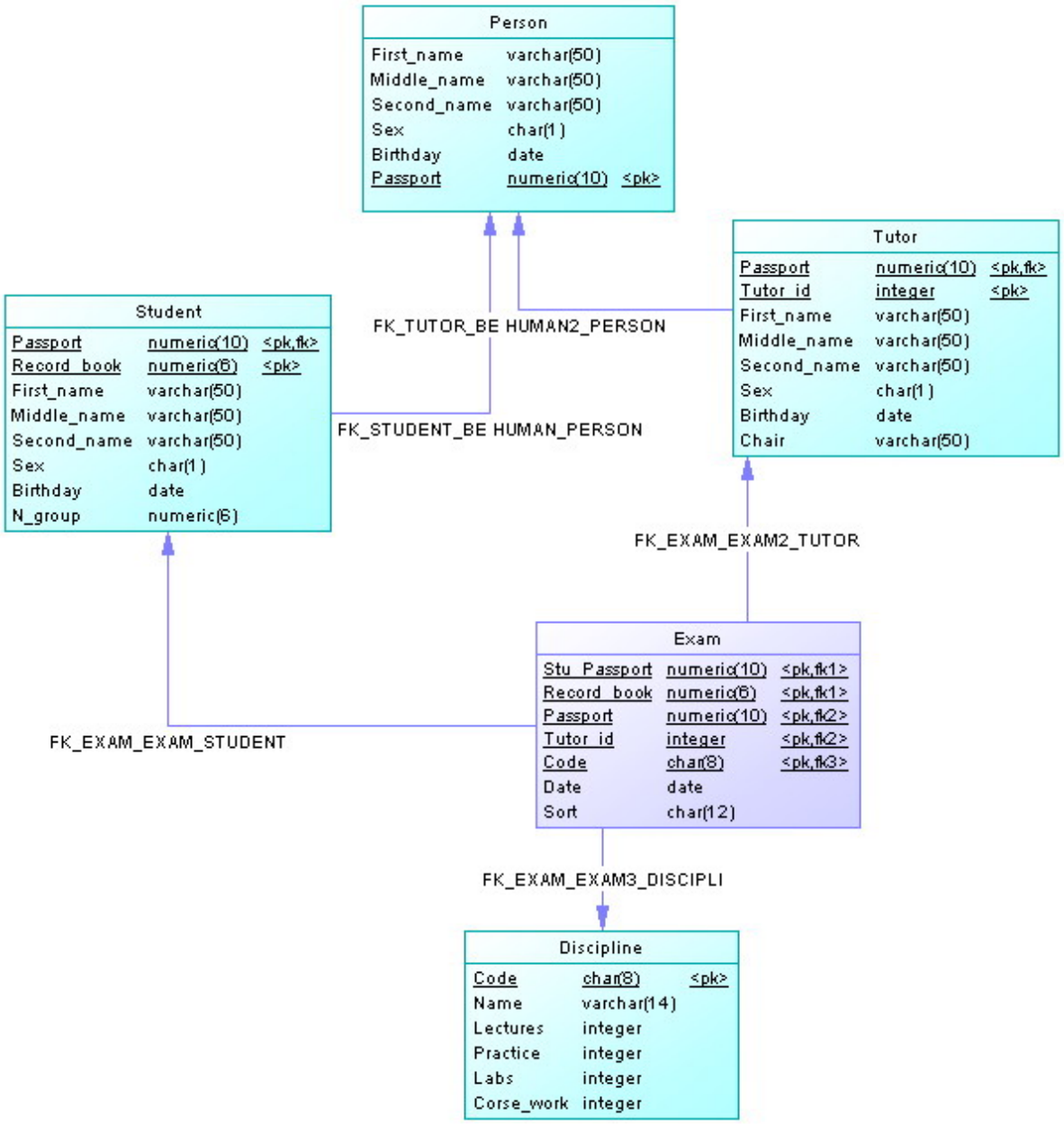


Рис. 2.5. Физическая модель базы данных для концептуальной модели на рис. 2.2.

Как видно из рис. 2.5., ассоциации в концептуальной модели соответствует таблица в физической модели. При этом среди атрибутов таблицы Exam есть явно избыточные. Так ключи Stu_Passport и Passport могут быть удалены из таблицы Exam.

Мы рассмотрели особенности концептуального моделирования, определяемые применением наследования и ассоциативных связей. Концептуальные модели такого типа, заданные диаграммами EERD, преобразуются

Диаграммы ERD, содержащие стандартные элементы концептуальных моделей баз данных - сущности и связи – преобразуются в физические модели обычным способом, с применением правил Джексона [5].

называется связь, обладающая атрибутами так же, как ими обладает сущность. Ассоциация может связывать несколько сущностей посредством бинарных связей, обладающих степенью и модальностью.

2.2. Проектирование и программирование приложений СУБД

Приложение - *это одна или несколько программ, работающих с базами данных через интерфейс пользователя. Как правило, приложение предназначено для решения задач, объединенных одной темой. Например, - это может быть бухгалтерское приложение, научное и т.п.*

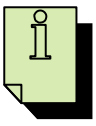
Термин «проектирование» применим к приложениям так же, как и к базам данных. Современные CASE – системы позволяют моделировать структуру и функции приложений, используя модели бизнес- процессов, а также набор объектно – ориентированных моделей, основанных на языке UML.

В курсовой работе необходимо выполнить проектирование графического интерфейса и программирование функций разрабатываемого приложения. Проектирование и программирование выполняется в системе Sybase PowerBuilder [6].

2.2.1 Проектирование интерфейса

При проектировании интерфейса целесообразно применять технологию окон данных, которая позволяет реализовывать комплексные решения для операций ввода - вывода и поиска данных.

Технология окон данных описана в [5, 6], поэтому здесь мы рассмотрим лишь некоторые характерные особенности применения данной технологии, которые обычно вызывают у студентов затруднения.



Дальнейший материал требует знакомства с технологией окон данных.

Синхронизация в окнах данных.

Иногда создается графический интерфейс, в котором применяется несколько контроллеров и, соответственно, объектов окон данных. Позиционирование на данные в одном из окон данных синхронно отображает соответствующие данные в других окнах.

На рис. 2.6. показан пример подобного интерфейса, в котором при выборе подразделения в окне «Отдел» отображаются соответствующие ему отделы, далее в окне «Сотрудники» - должности сотрудников выбранного отдела, и при выборе конкретной должности в правом окне отображаются персональные данные сотрудника.

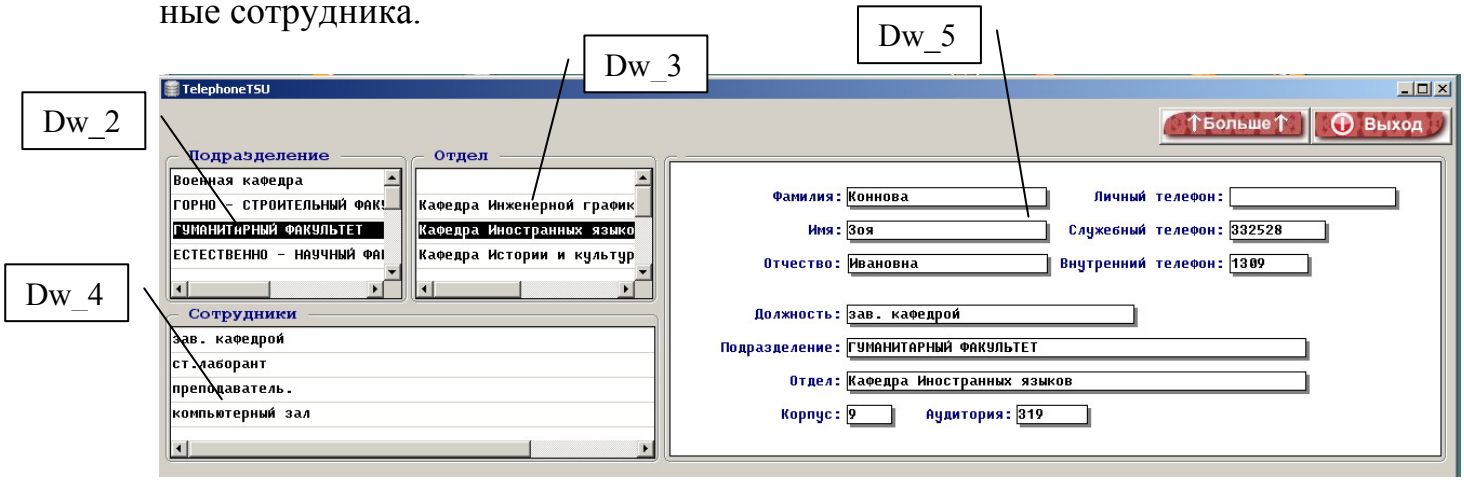


Рис. 2.6. Пример интерфейса с синхронизацией отображения данных в нескольких окнах.

Рассмотрим выполнение синхронизации.

Переход на определенную строку в окне, управляемом контроллером dw_2, инициирует событие rowfocuschanged. Скрипт – реакция на данное событие приведен на рис. 2.7.

```
1. string fakult, kaf
2. fakult=dw_2.GetItemString(dw_2.GetRow(),1)
3. dw_3.Retrieve(fakult)
4. setnull(kaf)
5. dw_2.Retrieve(kaf,fakult)
6. dw_2.ScrollToRow(1);
7. dw_5.Retrieve(dw_2.GetItemNumber(dw_2.GetRow(),2));
```

Рис. 2.7. Скрипт для события rowfocuschanged

В скрипте применяется функция GetItemString, позволяющая извлечь данные из заданного буфера окна данных. По умолчанию используется первичный буфер.

Таким образом, в строке 2 скрипта переменной fakult присваивается значение данных первичного буфера, извлекаемые из текущей строки и первого столбца. Здесь используется упрощенный синтаксис функции GetItemString (полный синтаксис см. в помощи PowerBuilder): первый аргумент – это номер строки буфера окна данных, второй аргумент – номер столбца. Номер строки - это номер текущей строки, определяемый как dw_2.GetRow(). Номер столбца постоянно равен 1, поскольку объект окна данных для контроллера dw_2 имеет лишь один столбец, в котором содержатся названия подразделений.

Далее в строке 3 скрипта применяется функция Retrieve(fakult) с параметром fakult. Это приводит к выводу в окно «Отдел» данных, соответствующих тому подразделению, которое определено текущим значением переменной fakult.

В других строках скрипта на рис. 2.7 данное решение повторяется.

2.2.2. Программирование функций приложения с использованием принципов объектно-ориентированного программирования

Как создать приложение в Power Builder

Создание простого приложения может состоять из следующих шагов, которые не обязательно выполнять строго в указанном порядке:

1. Создать базу данных или установить связь с уже существующей базой. Создать в базе данных необходимые таблицы.
2. Создать объект-приложение. Заодно создается библиотека, в которой будут сохраняться создаваемые далее классы объектов. В приложении можно определить набор шрифтов, используемых по умолчанию. В нем же будут сохраняться определения глобальных переменных прикладной системы, если таковые понадобятся.
3. Создать одно или несколько окон данных для обмена информацией с базой данных.
4. Создать головное окно прикладной системы, включив в него в качестве органов управления окна данных, командные кнопки и др., написать для окна и его органов управления программы обработки событий.
5. Написать программу события **open** для объекта-приложения. Эта программа может включать команды, необходимые для открытия базы данных, инициализацию глобальных переменных, команду открытия головного окна.
6. Скорее всего, необходимо создать меню и включить его в головное окно.

7. Наконец, после отладки прикладной системы, создается выполняемый модуль (.EXE), который может выполняться на компьютере, где не установлена система Power Builder.

Если создается более сложная прикладная система, то, необходимо воспользоваться и другими видами объектов, такими как ***структуры, функции, запросы, пользовательские объекты***. Предварительное продумывание структуры создаваемой системы поможет сэкономить программистские усилия, в частности за счет использования преимуществ объектно-ориентированного подхода, таких как ***наследование***.

Любое приложение, разрабатываемое в системе Power Builder, использует принципы объектно-ориентированного программирования (ООП).

Рассмотрим реализацию этих принципов на примере приложения «Система ввода данных в БД», которое доступно по ссылке [8]. В приложении решаются следующие задачи.

1. Использование универсального окна навигации (w_navigation) с элементом управления окнами данных и наследование от этого окна других окон навигации;
2. Использование пользовательского объекта (u_business_object) для инкапсуляции основных методов элемента управления окнами данных;
3. Использование объектов окон данных как для одной таблицы базы данных, так и для нескольких;
4. Обработка ситуации неподключения к источнику данных, а также разработка возможности выборочного подключения;
5. Обработка неправильного ввода данных.

Рассмотрим решения некоторых из этих задач.

Использование универсального окна навигации

Все операции ввода – вывода, реализуемые посредством технологии окон данных, используют одни и те же функции окна данных, но различные источники данных – таблицы базы данных. Поэтому целесообразно разработать один, единый интерфейс ввода – вывода, в котором в зависимости от конкретной ситуации менялись бы некоторые параметры, а основные решения оставались бы неизменными.

Такой интерфейс в виде универсального окна навигации (рис. 2.8.) был создан. От него наследуются другие объекты – окна, используемые для определённых целей (например, окно «Личные данные студентов»).

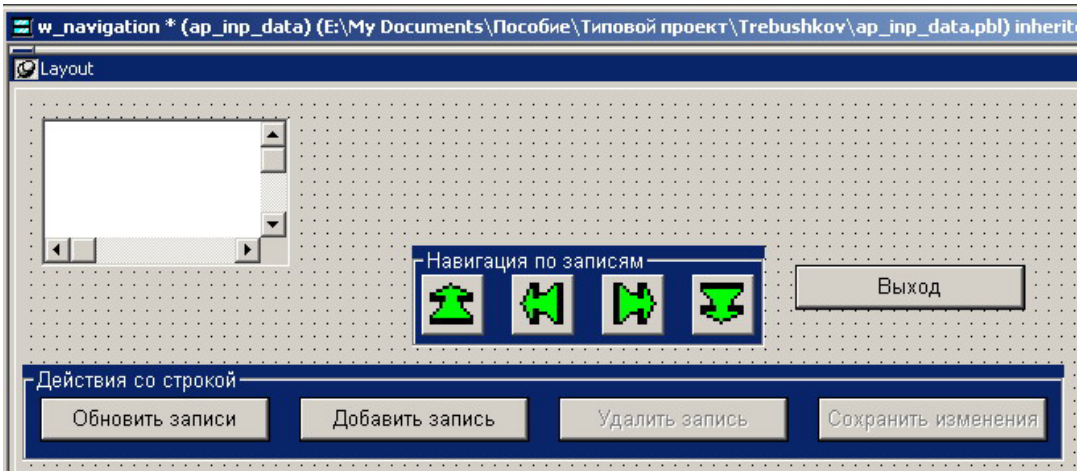


Рис. 2.8. Макет универсального окна навигации.

С универсальным окном навигации определены связанные с ним события, показанные на рис 2.9.

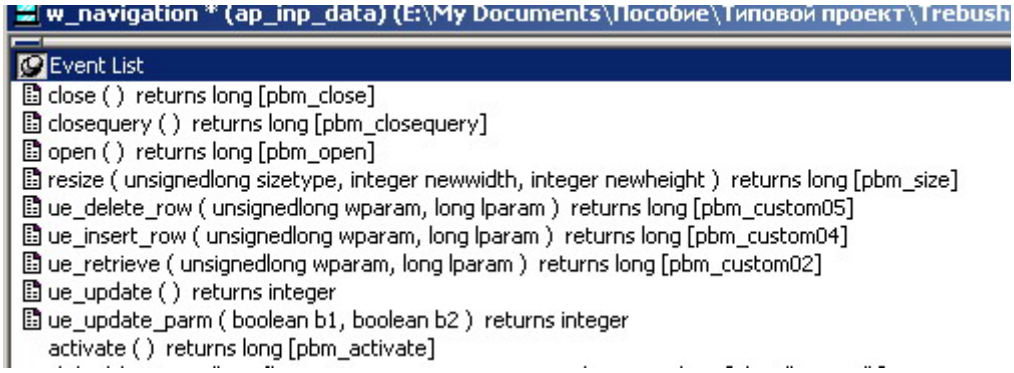


Рис 2.9. События универсального окна навигации

Данный объект использует другой, пользовательский объект u_business_object, который представляет собой набор специальных функций поддержки ввода – вывода. Этот объект и его функции показаны на рис. 2.10.

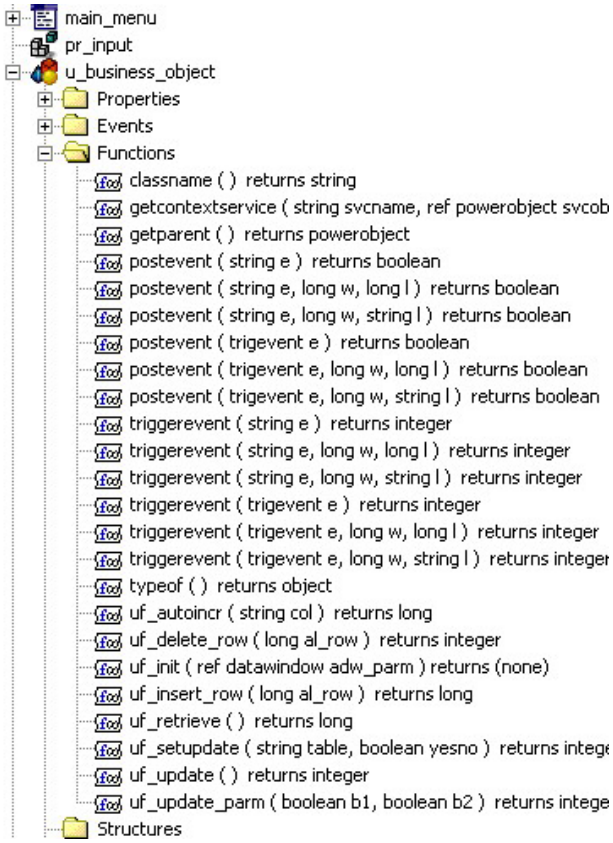


Рис. 2.10. Пользовательский объект u_business_object и его функции.

Далее определяем работу кнопок скриптами на события нажатия (clicked):

Например, для кнопки  (перейти к первой записи):

```
dw_1.ScrollToRow(1);
dw_1.setfocus()
```

Для кнопки «Выход»:

```
close(parent)
```

Для кнопки «Обновить»:

```
parent.TriggerEvent ("ue_retrieve")
```

Событие окна ue_retrieve (наряду с аналогичными событиями ue_update, ue_delete и др.) – пользовательское событие, определяемое скриптом:

```
Integer          li_RC
li_RC = iuo_business_object.uf_retrieve()
//Здесь обратились к функции объекта iuo_business_object
If li_RC < 0 Then
```

```
  cb_del.enabled = False
Else
  ib_changed = False
  cb_save.enabled = False
  If li_RC > 0 Then cb_del.enabled = True
End If
Return li_RC
```

Универсальный пользовательский объект

Объект iuo_business_object как экземпляр объекта u_business_object создаётся в секции внутренних переменных (Instance Variables) окна w_navigation (рис. 2.11.)

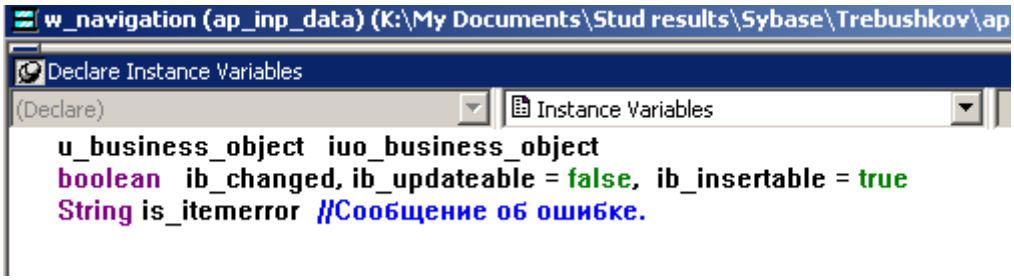


Рис. 2.11. Создание наследника пользовательского объекта.

А в окне – наследнике он может быть создан следующим образом:

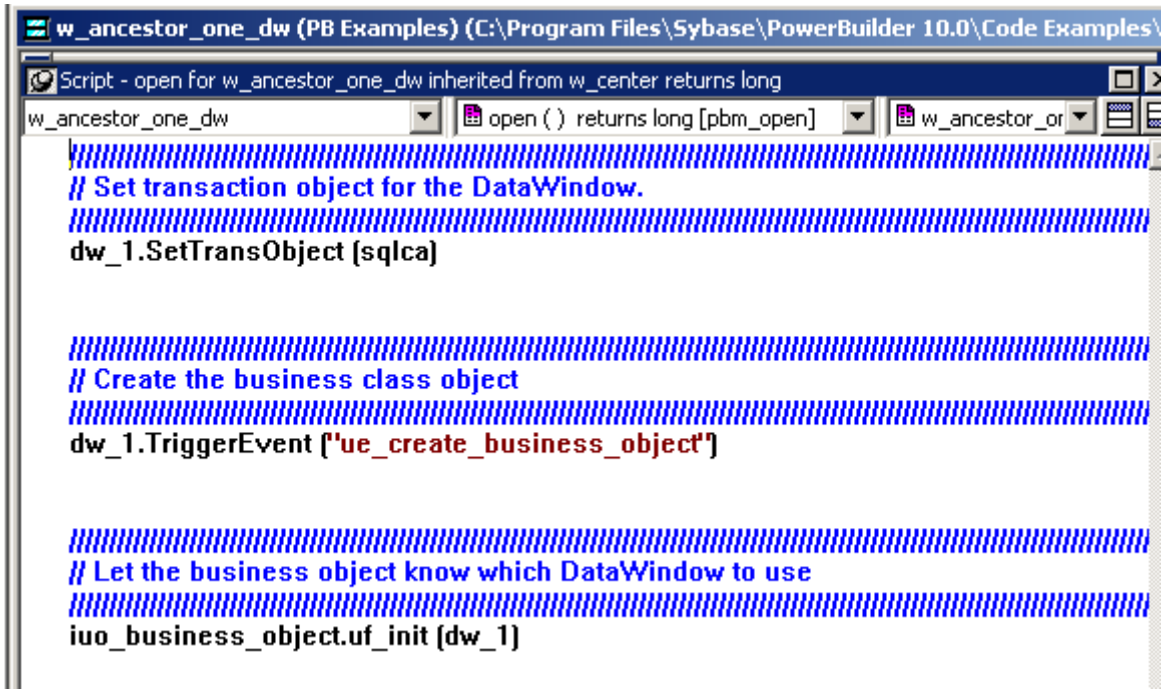


Рис. 2.12. Создание пользовательского объекта в окне – наследнике.

Обратите внимание на подкрашенность значков скрипта около событий – это признак того, что эти события наследуются от родительского объекта.

Здесь используем функцию TriggerEvent. Далее функция uf_init с аргументом dw_1 обеспечивает передачу ссылки на окно данных.

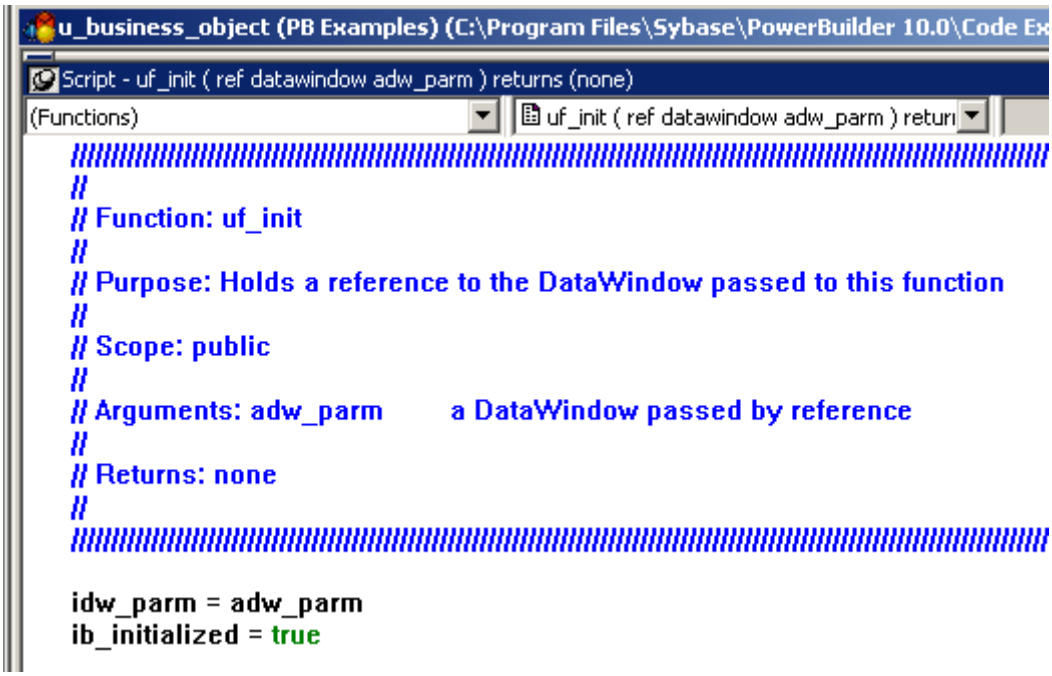


Рис. 2.13. Пример функции пользовательского объекта.

Это объект пользовательского класса. В этом класс собраны функции работы с окном данных.

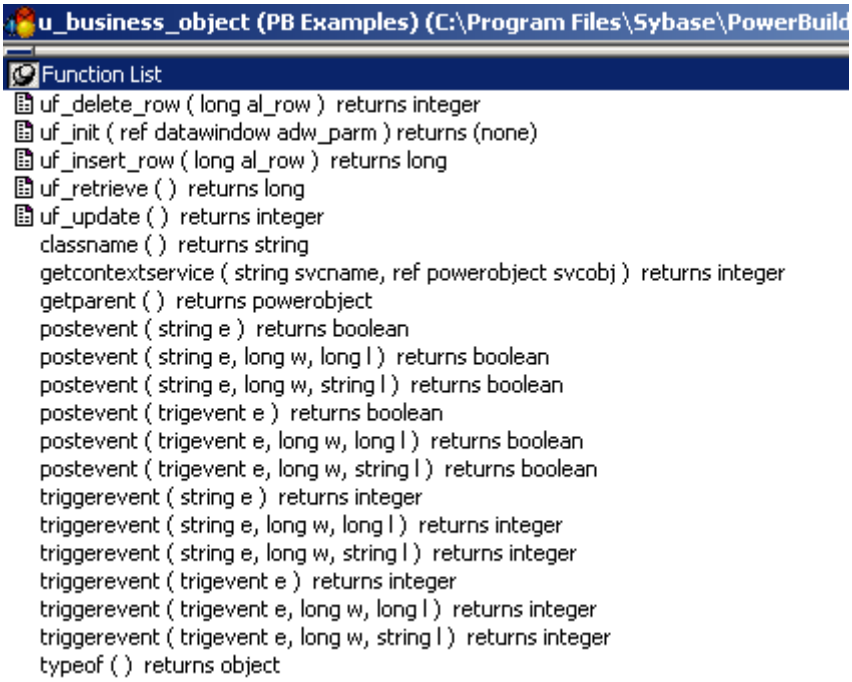


Рис. 2.14. Список функций пользовательского объекта.

Вот, например скрипт одной из функций пользовательского объекта (**uf_update**):

```
int      li_rc
li_rc = idw_parm.Update()
if li_rc = 1 then
        commit;
else
        rollback;
end if
     return li_rc
```

Объект **idw_parm** – это окно данных, создаваемое в объекте пользовательского класса **u_business_object**:

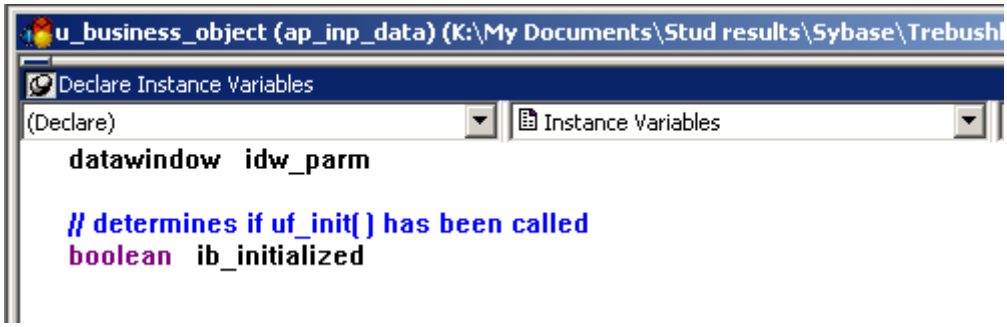


Рис. 2.15. создание объекта окна данных в пользовательском объекте.

Представленные здесь фрагменты программных решений обеспечивают универсальность этих решений. В приложении для решения новой задачи, использующей ввод-вывод данных, достаточно наследовать универсальное окно навигации и пользовательский объект.

3. ОФОРМЛЕНИЕ И ЗАЩИТА РАБОТЫ (ПРОЕКТА)

Курсовая работа оформляется согласно существующим стандартам [9].

К защите предъявляются:

- пояснительная записка к курсовой работе в печатном и электронном вариантах;
- программные компоненты (проект PowerBuilder), реализующие заданную СУБД.

Пояснительная записка должна содержать материалы, отражающие ход проектирования. К ним относятся:

- анализ задания,
- исследование вариантов и технологий реализации системы,
- описание алгоритмов работы приложения,
- описание интерфейсов системы,
- результаты испытаний (тестирования) системы.

Пояснительная записка к курсовой работе содержит следующие разделы.

Аннотация. Краткое изложение темы и содержания курсовой работы.

Введение. Во введении необходимо показать место данной работы в индустрии СУБД.

Анализ задания и выбор технологии решения задач. Поскольку курсовая работа выполняется на определенной платформе СУБД, необходимо привести информацию, демонстрирующую преимущества данной платформы. Особо уделить внимание средствам разработки приложений.

Проектирование базы данных. В данном разделе обязательно помещаются модель «сущность – связь» базы данных и результирующая физическая модель. Поясняется алгоритм построения физической модели.

Проектирование и программирование информационной системы. Данный раздел освещен выше.

Тестирование системы. Для подтверждения работоспособности созданной системы она должна быть продемонстрирована в работе. Среда Power Builder позволяет выполнить запуск приложения, существующего в виде проекта и не откомпилированного в двоичный модуль. При этом реализуются все режимы работы приложения, как если бы оно запускалось автономно в среде операционной системы. Такой способ демонстрации достаточен для проверки результатов проектирования. В особых случаях, когда создаются модули, включаемые в соз-

данные ранее программные продукты, необходима дополнительная работа по установке работоспособной версии всей системы.

В любом случае, на этапе демонстрации результатов проектирования студент демонстрирует свои практические навыки работы с информационными системам или, по крайней мере, компьютерную грамотность. От этого во многом зависит оценка его работы.

4. ЗАКЛЮЧЕНИЕ

В данном руководстве рассмотрены основные положения и примеры курсового проектирования информационных систем, выполняемого в рамках изучения дисциплины «*Базы данных*».

Для выполнения конкретного проекта данных указаний недостаточно. Необходимо воспользоваться литературой, приведенной в библиографическом списке.

Современной тенденцией в проектировании информационных систем является применение технологий автоматизированного проектирования. Подобные технологии относятся либо к CAD/CAM - Computer Aided Design (Manufacturing), что аналогично САПР, либо к CASE - технологиям, применяемым для автоматизированной разработки ПО.

Помимо автоматизации структурных методологий и, как следствие, возможности применения современных методов системной и программной инженерии, CASE обладают следующими основными достоинствами:

- улучшают качество создаваемого ПО за счет средств автоматического контроля (прежде всего, контроля проекта);
- позволяют за короткое время создавать прототип будущей системы, что позволяет на ранних этапах оценить ожидаемый результат,
- ускоряют процесс проектирования и разработки,
- освобождают разработчика от рутинной работы, позволяя ему целиком сосредоточиться на творческой части разработки;
- поддерживают развитие и сопровождение разработки;

К настоящему моменту дисциплина CASE оформилась в самостоятельное наукоемкое направление в программа - технике, повлекшее за собой образование мощной CASE-индустрии, объединившей сотни фирм и компаний различной ориентации. Среди них выделяются компании-разработчики средств анализа и проектирования ПО с широкой сетью дистрибьютерских и дилерских фирм; фирмы-разработчики специальных средств с ориентацией на узкие предметные области или на отдельные этапы жизненного цикла ПО; обучающие фирмы, ко-

торые организуют семинары и курсы подготовки специалистов; консалтинговые фирмы, оказывающие практическую помощь при использовании CASE-пакетов для разработки конкретных приложений; фирмы, специализирующиеся на выпуске периодических журналов и бюллетеней по CASE.

Эта технология вполне может применяться в курсовом проектировании, поскольку Лаборатория информационных систем кафедры располагает мощным CASE - средством проектирования реляционных баз данных Sybase PowerDesigner.

5. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Мейер Д. Теория реляционных баз данных - М.: Мир, 1987. - 608с.
2. Джексон Г. Проектирование реляционных баз данных для использования с микроЭВМ. - М.: Мир, 1991. - 252 с.
3. Советов Б.Я., Цехановский В.В., Чертовской В.Д. Базы данных. Теория и практика.. - М.: Высш. шк., 2007. – 463 с.
4. Нартова А. PowerDesiger 15. моделирование данных. - М., Лори, 21012. - 469 с.
5. Богатырев М.Ю. Разработка и программирование систем управления базами данных: Учеб. пособие. – Тула: изд-во ТулГУ, 2009. -145 с.
6. Смит Б.Дж., Шаад Г.У. Power Builder 5.0. Библия разработчика. - К.: Диалектика, 1997. -544 с.
7. Богатырев М.Ю. Введение в систему Power Builder . Методические указания к выполнению лабораторных работ. - Тула, изд-во ТулГУ, 1998. - 36 с.
8. Электронный ресурс: http://lis.tula.ru/Data/standard%20project%20PB8.rar
9. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе.
10. IEEE 1008 – 1986.Тестирование программных компонент