МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение
высшего профессионального образования
«Тульский государственный университет»

Кафедра «Автоматика и телемеханика»

СБОРНИК МЕТОДИЧЕСКИХ УКАЗАНИЙ К ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

БАЗЫ ДАННЫХ И ЗНАНИЙ

Направление подготовки: *230400 «Информационные системы и технологии»*

Профиль : *«Информационные системы»*

Формы обучения: *очная*

Тула 2012 г.

Методические указания к лабораторным работам 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 к лабораторным работам пересмотрены и утверждены на заседании кафедры *автоматики и телемеханики* факультета *кибернетики*,

протокол №___ от "___"_____ 20___ г.

Зав. кафедрой_____А.А. Фомичев

ЛАБОРАТОРНАЯ РАБОТА № 1

ЗНАКОМСТВО С СИСТЕМОЙ POWERBUILDER

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение принципов организации системы PowerBuilder и приобретение первоначальных практических навыков работы с системой

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СПРАВКА

Система PowerBuilder (PB) является одной из самых мощных реляционных СУБД профессионального уровня. Она представляет собой объектно-ориентированную среду разработки интерактивных прикладных систем, взаимодействующих с базами данных по технологии “клиент/сервер” и позволяет одному или нескольким разработчикам строить графические приложения, работающие с локальными, удаленными и распределенными базами данных. PowerBuilder отличается продуманным интерфейсом, удобной средой, широким выбором инструментов, открытостью, наращиваемостью и многоплатформенностью. PowerBuilder – это больше, чем просто инструментарий, – это среда разработки, в которой разработчик может не только создавать приложения, но и производить реструктурирование и любые модификации баз данных, а также перекачивать данные из одних источников в другие.

Графический пользовательский интерфейс приложений, создаваемых в PB, строится из обычных для MS Windows компонентов, таких как *окна, меню, органы управления – кнопки, поля редактирования, выпадающие списки* и др. Кроме того, для интерфейсов с базами данных имеется управляющий объект специального вида – так называемое *окно данных.* Посредством окон данных разработчик может организовывать разнообразные гибкие формы взаимодействия конечного пользователя с базой данных. Окна данных сочетают полные возможности генераторов отчетов, форм для ввода и просмотра данных, включают средства деловой графики; при этом они достаточно просты в изготовлении и использовании.

Построение даже довольно сложных пользовательских интерфейсов в среде PB требует относительно небольших трудозатрат, поскольку для каждого вида объектов – окон, меню, окон данных и др. – PB предоставляет специализированную среду разработки, так называемую мастерскую (Painter) с хорошо подобранным набором инструментов. При этом текущее состояние работы над объектом отображается в наглядной графической форме. Поведение объектов, составляющих прикладную систему, описывается на языке PowerScript. Имеются традиционные средства отладки.

2.1. Основные понятия и объекты среды Power Builder

Библиотеки

Каждое приложение представляет собой набор взаимодействующих программных модулей, реализующих определенные функции интерфейса системы, либо управления данными баз. Эти модули соответствуют объектам информационной системы, создаваемым при ее проектировании. Все эти объекты – меню, окна, органы управления, процедуры – помещаются в *библиотеку классов* PowerBuilder. Каждое приложение имеет свою библиотеку и может использовать объекты других библиотек.

Библиотека, в которой сохраняются описания классов объектов PB, представляет собой файл с расширением .pbl. Новая библиотека может быть автоматически создана в тот момент, когда создается новый объект-приложение. Описание каждого вновь создаваемого класса объектов попадает в библиотеку, которую необходимо указать. Описания классов

объектов, используемых прикладной системой, хранятся в одной или нескольких библиотеках. Эти библиотеки могут находиться на вашем компьютере или на сервере.

Художники

В PowerBuilder предусмотрены специализированные среды разработки для основных видов объектов – приложений, окон, меню и др. Эти среды мы назовем *художниками,* их оригинальное название painter.

В большинство художников можно попасть при помощи кнопок на главной панели инструментов. В некоторые вспомогательные художники можно попасть только из других художников.

Одновременно может быть открыто несколько художников. Для переключения между открытыми в данный момент художниками используйте меню Window.

Выход из художников может осуществляться при помощи меню File/Close, нажатием Ctrl+F4.

Рабочая область

В ранних версиях PowerBuilder можно было единовременно работать только с одним приложением. Версия PowerBuilder 8.0 позволяет работать с несколькими приложениями, включив их в одну рабочую область (workspace).

Правило правой кнопки (всплывающие меню)

Это правило действует почти во всех художников. Нажатие правой кнопки мыши после подведения курсора к изображению какого-либо объекта в рабочем пространстве мастерской, вызывает относящееся к нему всплывающее меню. Например, в художнике окон (Window Painter) таким образом вызывается всплывающее меню для органов управления или самого окна (если нажать правую кнопку мыши на свободном пространстве окна). Всплывающее меню позволяет просмотреть или переустановить значения атрибутов объекта или выполнить такие действия, как уничтожение, копирование, вход в мастерскую программиста.

Получение помощи во время сеанса

Меню Help, кнопка Help на главной панели инструментов.

В мастерской программиста вы можете получить контекстно-зависимую справку о функции или ключевом слове. Выделите интересующее вас слово или имя функции и нажмите Shift-Fl.

Текущий объект

Довольно часто, перед тем как выполнить какое-либо действие над объектом, например, уничтожить его нажатием клавиши Delete, необходимо сделать его текущим. Это достигается нажатием левой кнопки мыши на объекте. Изображение текущего объекта (органа управления в окне, таблицы базы данных) выделено на экране либо цветом, либо четырьмя черными точками в углах.

Если нужно выделить несколько объектов, например, несколько кнопок в окне для их последующего выравнивания по размеру или положению, то для этого есть два способа:

- нажмите левую кнопку мыши, подведя курсор к первому объекту; затем нажмите на клавиатуре клавишу Ctrl и, не отпуская ее, помечайте мышью остальные объекты;

- если объекты находятся близко друг от друга, так, что их можно окружить прямоугольником, то поместите указатель мыши в один из углов этого воображаемого прямоугольника, нажмите левую кнопку мыши и, не отпуская ее, ведите указатель мыши к его противоположному углу. На экране будет виден пунктирный прямоугольник. Когда он окружит все желаемые объекты, отпустите кнопку мыши.

Буксировка объектов

Для перемещения объекта подведите к нему курсор мыши, нажмите левую кнопку, и, не отпуская ее, передвигайте объект в нужное положение.

Уничтожение объектов

Нажатие Delete приводит к уничтожению текущего объекта с предварительным запросом подтверждения.

Как создать приложение в PowerBuilder

Создание простой прикладной системы 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 в виде библиотеки, 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.
3. Создать одно или несколько окон данных для обмена информацией с базой данных.
4. Создать *главное 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. При необходимости создать меню и включить его в главное окно.
7. После отладки прикладной системы, создается выполняемый модуль (.EXE), который может выполняться на компьютере, на котором не установлена система Power Builder.

Для создания простейшего приложения в PowerBuilder следует использовать инструкции Приложения 1 к лабораторной работе N 1.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

- Запустить PowerBuilder
- Создать новую рабочую область (Workspace)
- Создать целевой объект (Target)
- Создать главное окно приложения (Window)
- Создать сценарий для события open приложения
- Запустить приложение

ЛАБОРАТОРНАЯ РАБОТА № 2

ВЗАИМОДЕЙСТВИЕ С БАЗАМИ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Целью работы является приобретение практических навыков работы с базами данных в системе PowerBuilder

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СПРАВКА

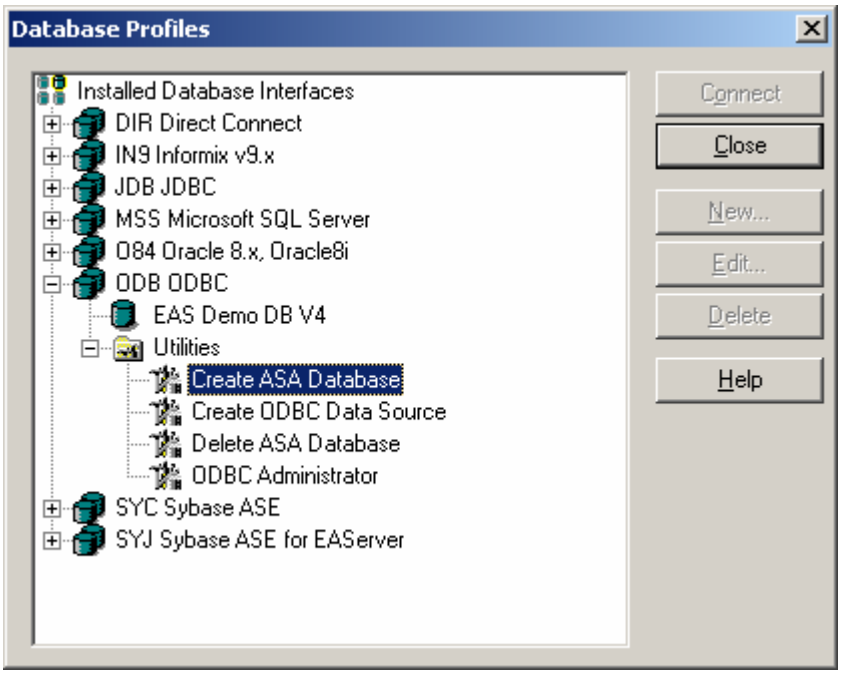
2.1. Базы данных PowerBuilder

Система Power Builder работает как с базами данных собственного формата *.DB, так и с базами данных других форматов. К ним относятся практически все известные форматы баз данных, например, dBASE, Excel, FoxPro, Paradox, Oracle, Informix. Такая универсальность достигается за счет применения *единого механизма подключения и работы с базами данных.* Независимо от форматов, все современные базы данных используют такие понятия как

- таблица;
- запись;
- поле;
- тип данных в поле;
- ограничения на данные в поле;
- индекс;
- ключ.

При подключении к базе данных используются понятия *источник данных* и *профиль.*

Для создания новой базы данных следует использовать пункт Create ASA Database окна Database Profile.



2.2. Мастерская баз данных

В рабочем пространстве мастерской баз данных можно создавать новые таблицы, просматривать условные изображения таблиц, их ключей и индексов. Нажатие правой кнопки мыши на имени таблицы, столбца, изображении ключа или индекса приводит к

появлению всплывающего меню для соответствующего объекта. Вызов мастерской баз данных осуществляется нажатием кнопки  панели инструментов.

Чтобы создать в базе данных новую таблицу, достаточно щелкнуть правой кнопкой мыши по строке Tables в левой части мастерской баз данных и выбрать пункт New Table.

Для того, чтобы изменить структуру уже созданной таблицы, следует выбрать ее имя в списке таблиц, щелкнуть правой кнопкой мыши и выбрать пункт меню Alter Table.

Любой из вышеперечисленных способов приведет вас к дизайнеру таблицы.

Создание таблицы начинается с определения имен и типов полей. Имя поля таблицы вводится в столбце Column Name. В следующем столбце устанавливается тип поля таблицы. В колонке Width можно задать длину поля, если только эта длина жестко не определена. Колонка таблицы Null – свойство поля Allow Nulls. Это очень важное свойство. Оно определяет, может ли поле принимать значение NULL. Если при добавлении записи в таблицу значение поля не определено, то ему присваивается значение NULL (если Allow Nulls=true).

2.3. Управление изображением таблиц

Текущий элемент рабочего пространства. Для того, чтобы сделать текущей таблицу, щелкните мышью над именем. Имя текущей таблицы выделяется инверсным изображением. Так же можно сделать текущим ключ или индекс.

Перемещение таблиц в рабочем пространстве. Таблицы перемещаются мышью за среднюю часть заголовка.

2.4. Связывание таблиц

Определение ***первичного ключа*** гарантирует уникальность данных в соответствующем столбце (или столбцах) таблицы. Первичные ключи служат основой для определения внешних ключей. В таблице может быть только один первичный ключ, который единственным образом идентифицирует каждую строку в таблице. Он может быть составным, но его значения должны быть уникальны. В мастерской баз данных Power Builder первичный ключ обозначается 

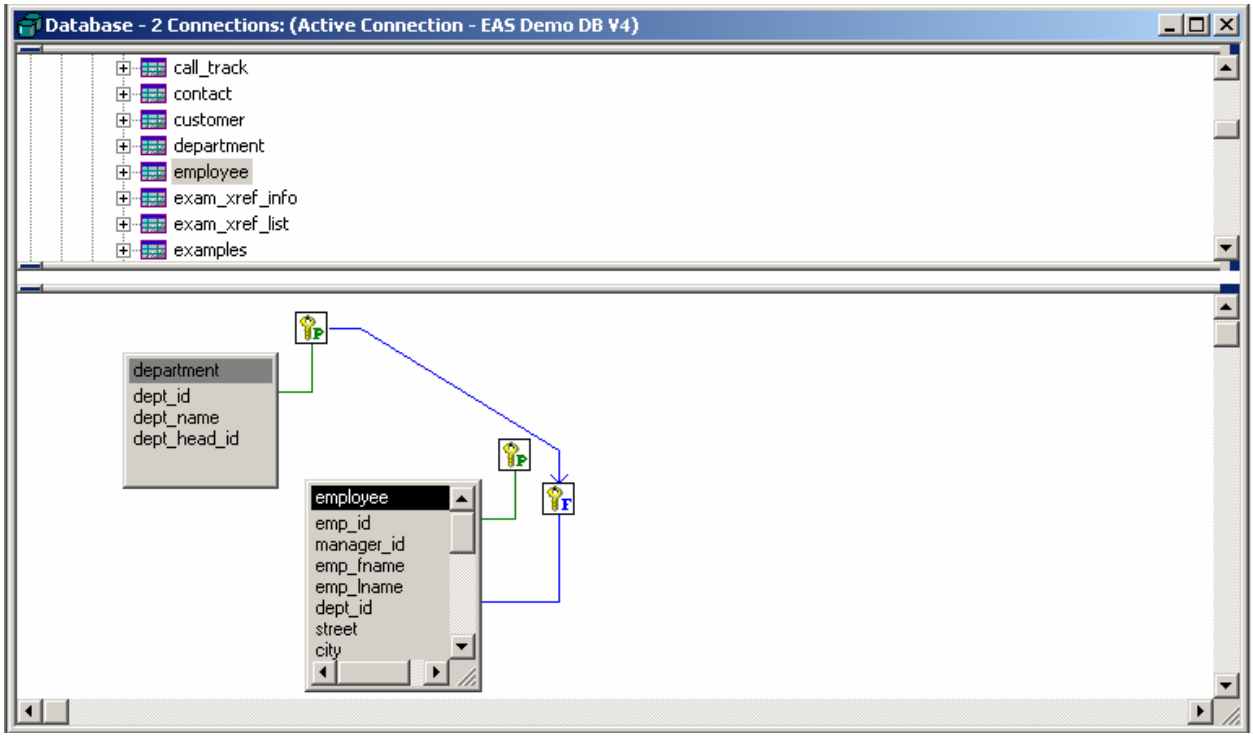
Внешние ключи описывают связи (соединения) между таблицами базы данных.

Например, в таблице employee определен внешний ключ dept_id, связывающий одноименный столбец с первичным ключом таблицы department. Описание этого внешнего ключа гарантирует, что при выполнении SQL-команд INSERT или UPDATE невозможно занести в столбец dept_id таблицы employee элемент данных, который бы не совпадал с ключом одной из записей таблицы department. В мастерской баз данных Power Builder внешний ключ обозначается  .

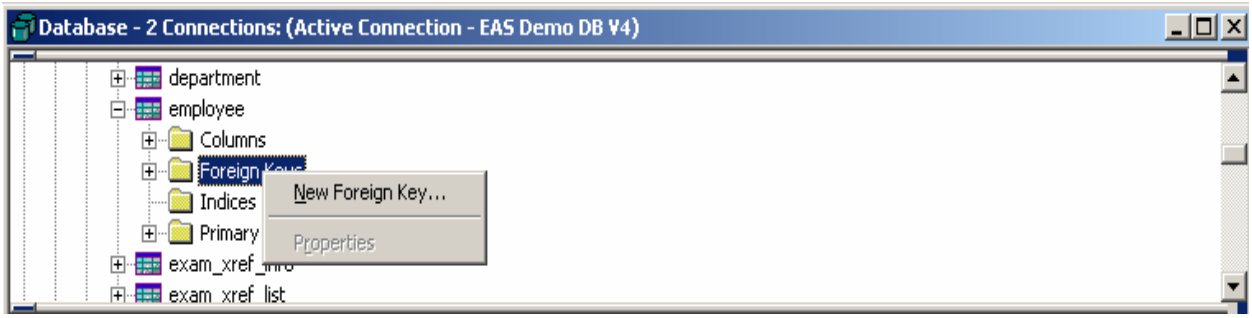
Правила:

1. *Внешний ключ должен иметь столько же полей, что и первичный ключ родительской таблицы*
2. *Типы полей первичного и соответствующего внешнего ключа должны совпадать.*

Это означает, что при связывании таблиц мы «экспортируем» элементы первичного ключа в другую таблицу, но не требуем уникальности значений внешнего ключа.



Для задания первичного (внешнего) ключа таблицы в мастерской баз данных Power Builder следует открыть выпадающий список свойств таблицы и, нажав правую кнопку мыши на пункте списка Primary Keys (Foreign Keys), выбрать New Primary Key (New Foreign Key), как показано ниже.



2.5. Ввод данных

Для ввода данных в базу данных следует открыть выпадающий список свойств таблицы, в которую требуется поместить данные, и выбрать пункт Edit Data.

2.6. Подключение к базе данных

Предположим, что у вас есть база данных ODBC, к которой еще не выполнялось подключение с помощью Power Builder на данном компьютере.

Прежде, чем подключать Power Builder к базе данных, необходимо выполнить несколько подготовительных операций.

1. Создать профиль базы данных (database profile).

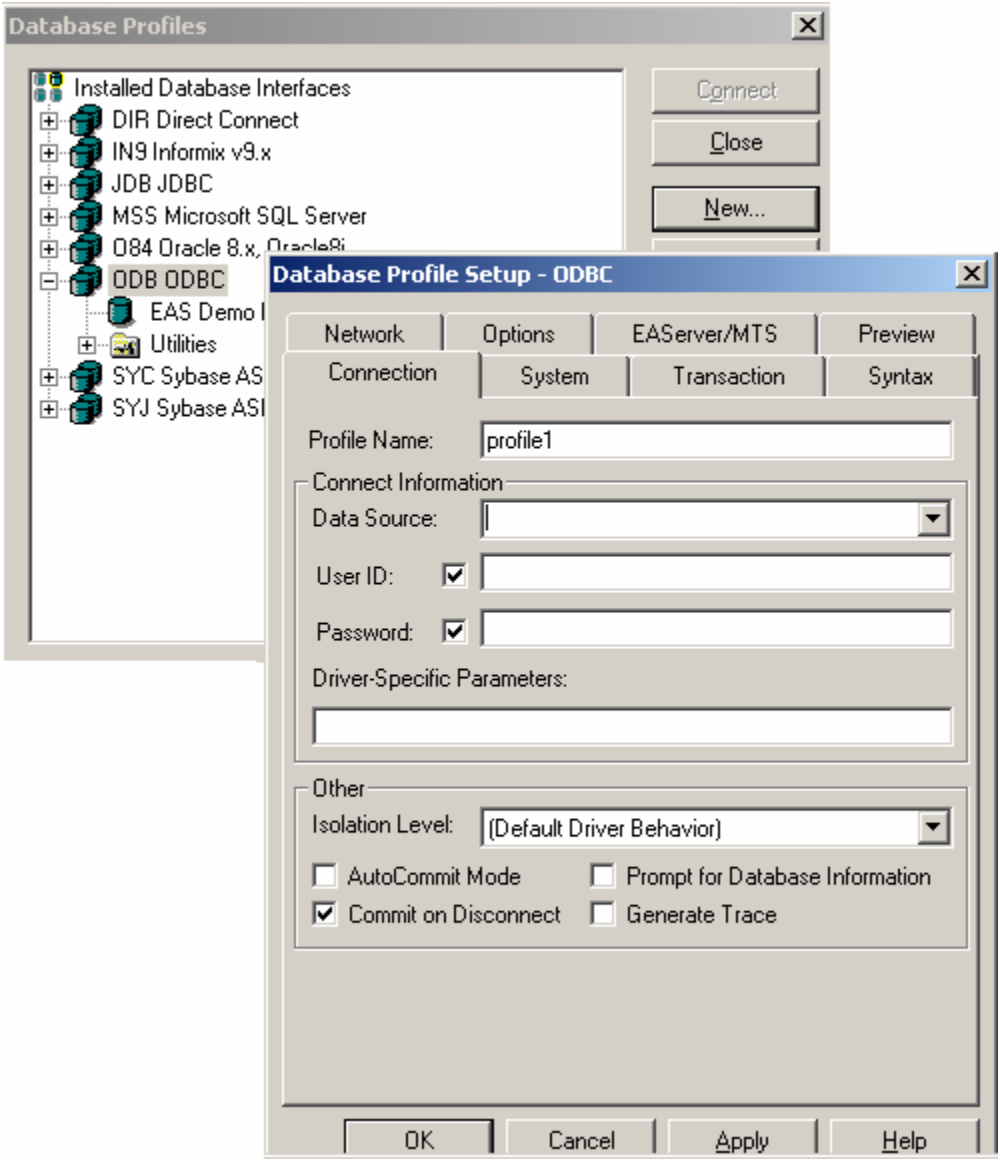
Если вы подключаетесь к базе данных ODBC, то необходимо выполнить следующий шаг.

2. Настроить конфигурацию источника данных. (data source configuration).

Профиль базы данных – это набор параметров, идентифицирующих базу данных. Профиль базы данных предоставляет Power Builder всю необходимую информацию для подключения к базе данных.

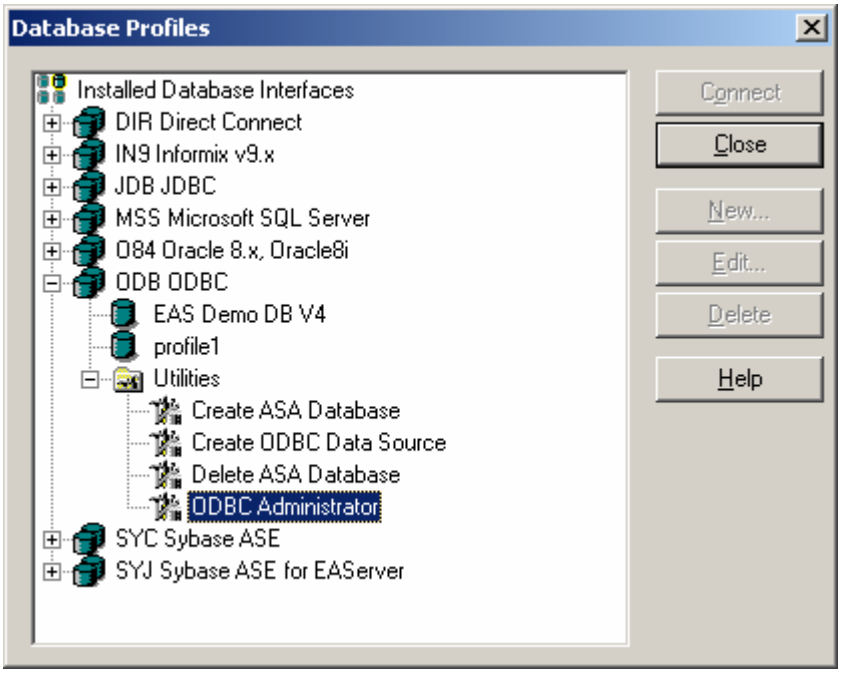
Профиль создается из системы Power Builder. Для открытия окна Database Profiles (Профили баз данных) следует нажать кнопку DB Profile  .

При нажатии кнопки New будет открыто диалоговое окно Database Profile Setup (Установка профиля баз данных), текстовые поля которого необходимо заполнить.



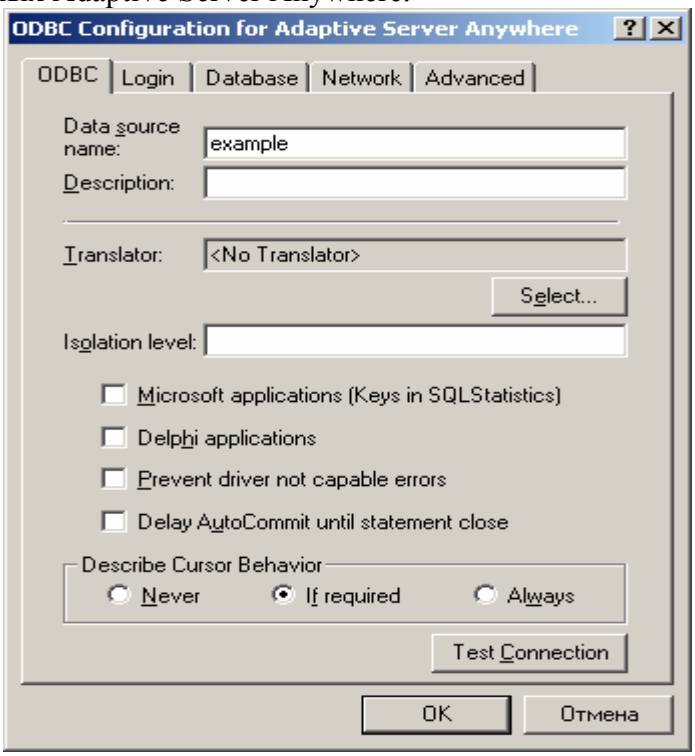
В диалоговом окне Database Profile Setup содержимое текстового поля Profile Name (Имя профиля) является уникальным идентификатором базы данных для Power Builder и только по этому имени Power Builder будет обращаться к этой базе данных. Имя базы данных в профиле необязательно должно соответствовать действительному наименованию базы данных.

Обратим внимание на поле Data Source (Источник данных). Для того, чтобы получить доступ к базе данных, необходимо создать конфигурацию источника данных. В окне Database Profile выберите пункт ODBC Administrator (Администратор источников данных ODBC).

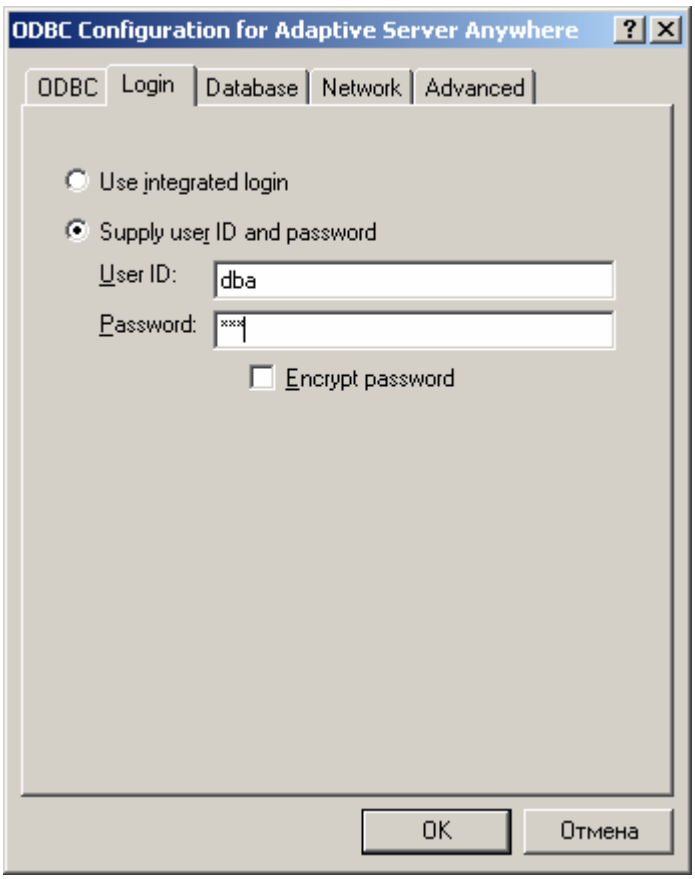


В появившемся диалоговом окне будет выведен список всех драйверов ODBC, инсталлированных на данном компьютере, и имена конфигураций источников данных, существующих для этого драйвера.

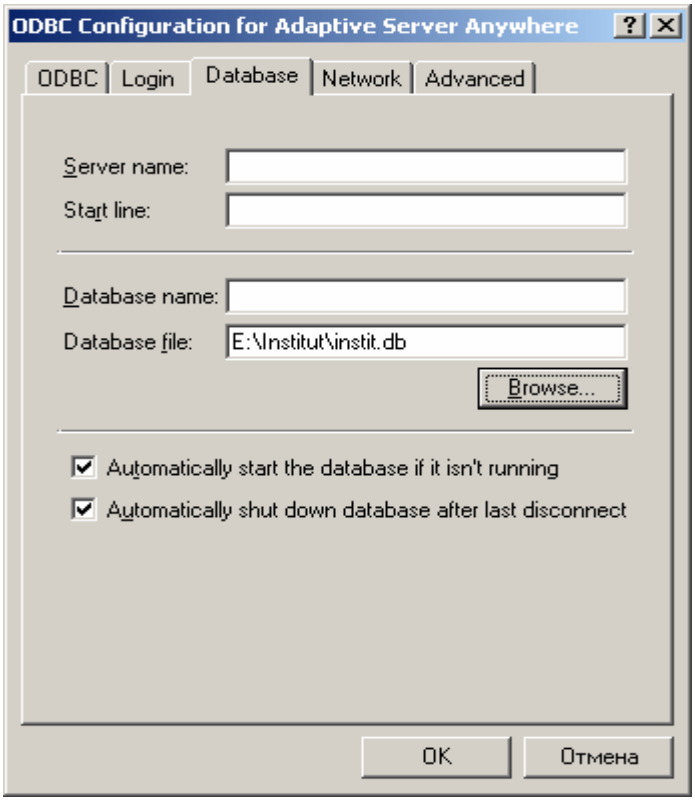
Чтобы создать новую конфигурацию источника данных, следует нажать кнопку Add (Добавить), выбрать драйвер и в диалоговом окне ODBC Configuration указать требуемые параметры. Для каждого драйвера существует свое диалоговое окно. Ниже приведен пример окна для источника данных Adaptive Server Anywhere.



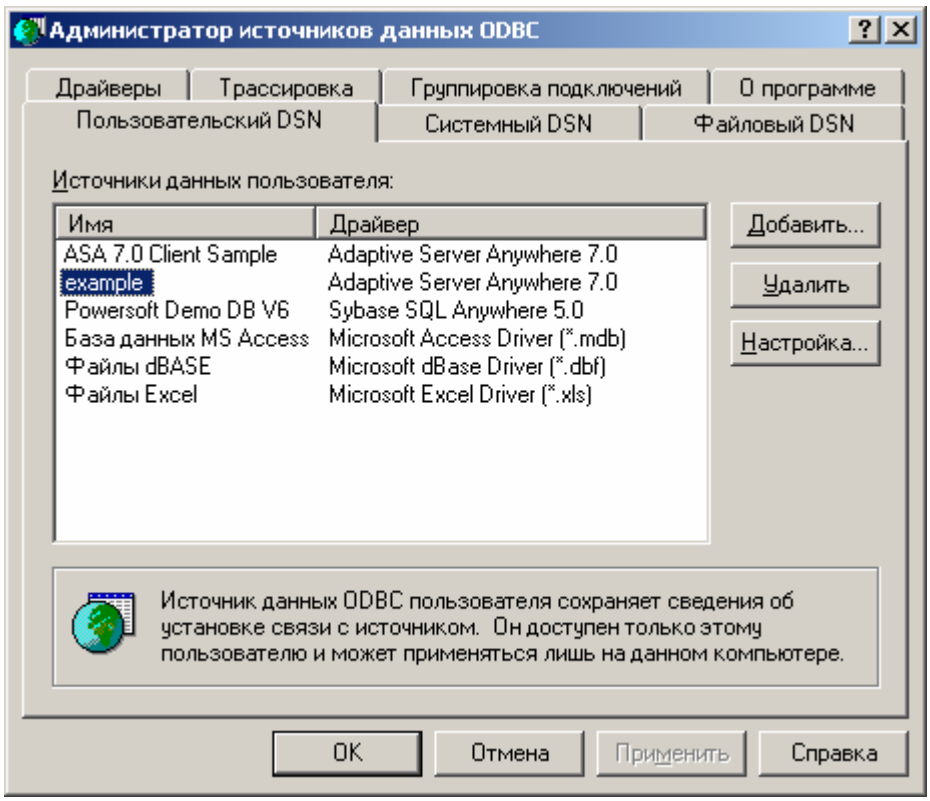
В закладке ODBC данного окна следует заполнить текстовое поле Data Source Name, указав имя источника данных. Затем в закладке Login в текстовые поля User ID и Password следует ввести имя пользователя базой данных и пароль. При создании базы данных Power Builder по умолчанию задает имя пользователя dba и пароль sql.



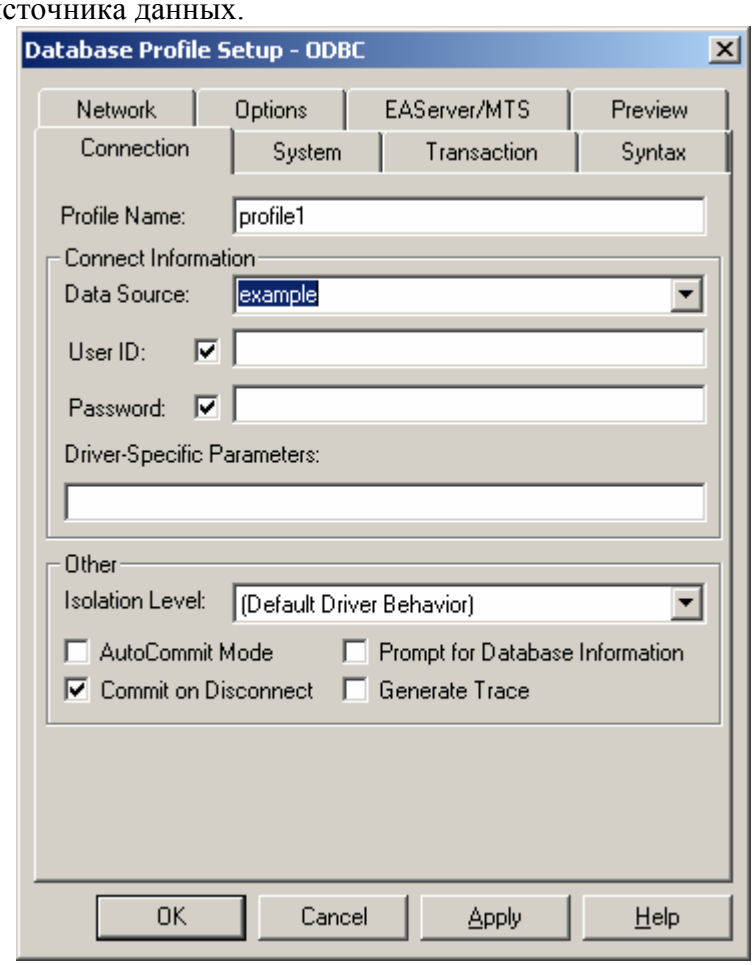
После этого в закладке Database следует заполнить поле Database file, указав путь к файлу базы данных.



После нажатия кнопки OK к списку существующих источников данных будет добавлен новый.



После создания новой конфигурации источника данных в окне Database Profile Setup следует указать имя источника данных.



Сохраните все изменения и закройте диалоговое окно. Теперь подключиться к базе данных можно с помощью кнопки Connect окна Database Profiles или по щелчку мыши на

имени профиля в мастерской баз данных, вызываемой с помощью кнопки  панели инструментов.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

3.1. Запустить PowerBuilder.
3.2. Создать новую базу данных.
3.3. Открыть мастерскую баз данных и создать 3 таблицы, описать связи между таблицами.
3.4. Ввести данные в созданные таблицы.

4. ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать наименование работы, цель работы, задание на работу, схему базы данных, описание полей каждой из созданных таблиц, примеры данных, которые могут храниться в таблицах (не менее 3-х записей для каждой таблицы), вывод.

ЛАБОРАТОРНАЯ РАБОТА N 3
РАЗРАБОТКА ПРИЛОЖЕНИЙ В СРЕДЕ POWERBUILDER. ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА

1. ЦЕЛЬ РАБОТЫ

Целью работы является приобретение практических навыков проектирования интерфейса информационных систем средствами системы PowerBuilder.

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СЦРАВКА

Каждая информационная система уникальна. Тем не менее, информационные системы всегда имеют определенные общие черты. Если разные приложения работают в единой среде, например Windows, они должны иметь типовой оконный интерфейс Windows. Другим фактором, определяющим общие черты приложений, является совокупность единых приемов работы с интерфейсом, сложившихся как результат опыта эксплуатации конкретных информационных систем. К типовым задачам, решаемым в интерфейсе большинства систем, относятся следующие задачи:

- Навигация в базе данных: просмотр записей, переход в начало таблицы, в конец таблицы, на конкретную запись;
- Ввод данных в базу;
- Поиск в базе данных, отвечающих заданным условиям;
- Вывод отчетной информации.

Органы управления создаются в мастере окон (Window Painter). Автоматически созданному в окне объекту присваивается имя, состоящее из сокращенного названия типа объекта и порядкового номера в окне, например, cb_3 – это третья командная кнопка (Command Button) среди всех командных кнопок окна. Имена объектов используются далее при создании скриптов, поэтому обычно их изменяют для того, чтобы отразить в имени смысл объекта.

Для вызова мастера окон требуется нажать кнопку New панели инструментов и выбрать пиктограмму Window закладки PB Object, как показано на рисунке 1.

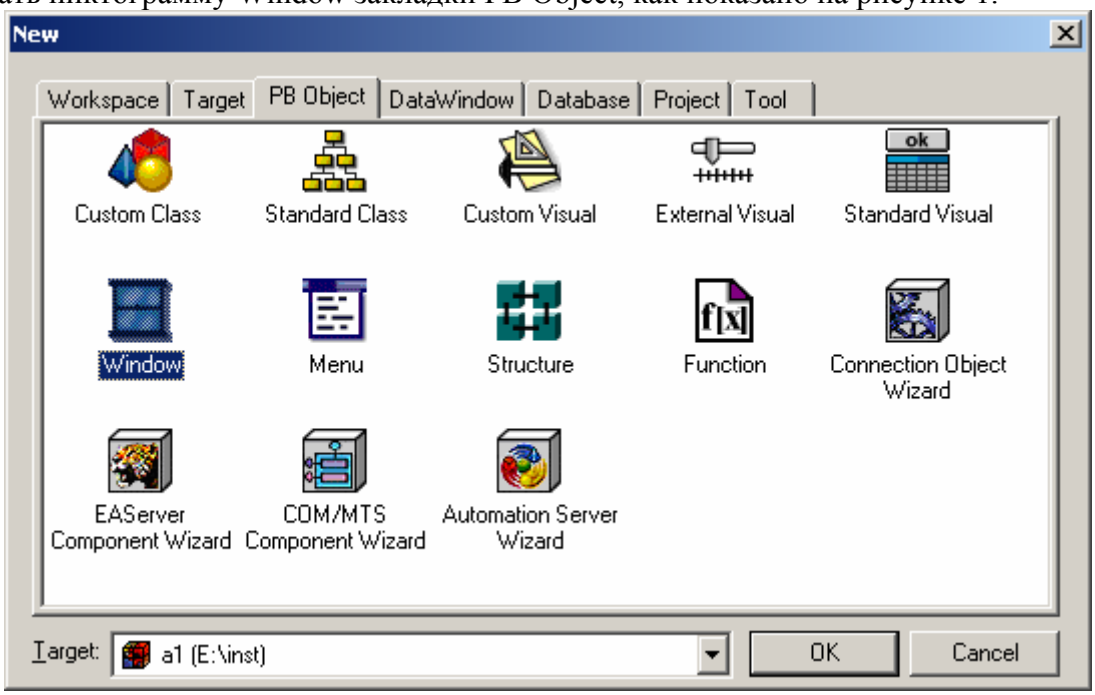


Рис.1. Вызов мастера окон

Аналогичным образом происходит вызов мастера меню (пиктограмма Menu), в котором создаются элементы меню.

Создаваемое приложение может включать в себя произвольное количество окон, точно так же, как оно может содержать любое количество меню.

При создании элемента меню, имя, которое вводится в поле Name представляет собой имя объекта элемента меню, которое далее может быть использовано при создании скриптов. По умолчанию Power Builder присваивает имя добавлением префикса “m_” к введенному тексту. Например, для опции File вводится имя “m_file”. В случае изменения введенного текста Power Builder соответственно изменит присвоенное имя. Изменение имени, однако, может привести к путанице в существующих сценариях, поэтому рекомендуется устанавливать флажок Lock Name (Блокировка имени), чтобы PowerBuilder в дальнейшем не смог изменить имя элемента меню.

Для того, чтобы включить меню в окно требуется вызвать мастер окон, открыть страницу свойств окна и на вкладке General (общие свойства) в текстовом поле Menu Name (имя меню) ввести имя меню.

При выполнении приложения и открытии окна вместе с ним появится и содержащееся в нем меню.

Существует шесть типов окон:

- главное (main);
- дочернее (child);
- раскрывающееся (popup);
- окно ответа (response);
- рамка MDI (mdi);
- рамка MDI с MivcroHelp (mdihelp).

Еще один тип окон – страница. Страницей называется главное, раскрывающееся, дочернее окно или окно ответа, открытое внутри рамки MDI.

В мастере окон тип окна задается на вкладке General страницы его свойств.

Разделяют родительские и дочерние окна. Родительское окно открывает дочернее.

Иногда дочернее окно полностью управляет приложением. Такое окно называется модальным. Пока открыто модальное окно, все остальные окна в приложении недоступны, хотя и могут быть видимыми. В PowerBuilder единственные модальные окна – это окна ответа. Все диалоговые окна обычно являются ответными. В PowerScript есть функция MessageBox для создания простых диалоговых окон. Если нужное диалоговое окно можно получить с помощью MessageBox, то нет необходимости конструировать отдельное окно. Но в случае, если окно должно поддерживать более сложные взаимосвязи с пользователем, например, запрос диапазона дат для получения выборки отчетов, то понадобится создать окно ответа.

Большинство окон в приложении должно быть главными: окна с DataWindow, окна с кнопками, страницы в рамке MDI. Если приложение состоит из одного окна, оно должно быть главным.

Раскрывающееся окно похоже на главное, но оно всегда располагается сверху своего родительского окна, даже когда оно не активно. Оно может оказаться полезным для использования в качестве страницы с инструкциями о последовательности действий, которые пользователь выполняет в главном окне, расположенном под ним.

В отличие от раскрывающихся окон, дочернее окно всегда остается внутри рамки своего родительского окна.

Рамка MDI – это окно, предназначенное для того, чтобы содержать дочерние окна, называемые страницами. MDI означает Multiple Document Interface (Многооконный интерфейс).

Термин MicroHelp относится к строке состояния, в которой показываются сообщения. Она располагается внизу окна.

В случае, если окно имеет тип “рамка MDI”, в него можно добавить панель инструментов, являющейся частью меню в мастере меню. Каждая кнопка панели инструментов должна быть эквивалентна соответствующей опции меню. Для создания панели инструментов используется вкладка Toolbar страницы свойств элемента меню.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

1. Запустить PowerBuilder.
2. Создать новое приложение.
3. Создать оконный интерфейс приложения для работы пользователя с созданной в лабораторной работе N 2 базой данных.

4. ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать наименование работы, цель работы, задание на работу, оконный интерфейс приложения с описанием назначения каждого окна и элементов управления, вывод.

ЛАБОРАТОРНАЯ РАБОТА N 4

РАЗРАБОТКА ПРИЛОЖЕНИЙ В СРЕДЕ POWERBUILDER. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД

1. ЦЕЛЬ РАБОТЫ

Целью работы является приобретение практических навыков применения метода объектно-ориентированного программирования в системе PowerBuilder.

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СЦРАВКА

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.

Проектирование приложения при помощи PowerBuilder затрагивает следующие вопросы:

- доступ к данным: через сети или локально: *таблицы и БД*;
- пользовательский интерфейс: *окна и органы управления*;
- управление выполнением (processing) приложения: *события и скрипты*;
- расширение функциональности приложения: *внешние программы*;
- стандартные выходные данные приложения: *отчеты, файлы, каналы (pipelines)*.

PowerBuilder – это объектно-ориентированная система, в которой полностью поддерживаются такие возможности, как *многоуровневое наследование*, *полиморфизм*, *инкапсуляция*. При этом концептуальная глубина сочетается со стремлением сделать интерфейс как можно более простым для освоения и использования. От начинающего разработчика “маскируется” сложность многих понятий, связанных с объектно-ориентированным подходом. Тем не менее, представляется важным с самого начала определить некоторые из этих понятий. Дело в том, что, выполняя дизайн приложения, разработчик каждый раз реализует принципы объектно-ориентированного подхода.

Объектно-ориентированное программирование (ООП) – система программирования, основанная на применении абстракции данных, модульной иерархии типов данных и свойствах:

- полиморфизм (свойство метода с одним и тем же названием обладать различным содержанием применительно к разным классам)
- наследование (свойство подкласса обладать характеристиками породившего его класса)
- инкапсуляция (свойство содержать в скрытом виде информацию, присущую объекту: структуру данных, фрагмент кода).

В основе системы объектно-ориентированного программирования Power Builder лежит объектная модель. Ниже описаны основные элементы объектной модели Power Builder.

Объекты PowerBuilder

Объект PowerBuilder – это сущность, обладающая набором атрибутов (свойств), набором методов (функций) и способная обрабатывать (реагировать на) некоторый набор событий.

Атрибуты

Атрибут – элемент данных, влияющий на внешний вид и/или поведение объекта.

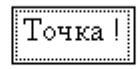
Примеры:
BackColor – атрибут типа long, определяет цвет окна.
Visible – атрибут любого графического объекта, имеет тип boolean; значение false означает, что объект невидим.
Control – атрибут окна, массив элементов типа PowerObject, содержит органы управления данного окна.
Как только какой-либо атрибут объекта получает новое значение, это тотчас же отражается на его внешнем представлении и поведении.
Для каждого вида объектов в PowerBuilder предопределен набор атрибутов с заданными начальными значениями. Определяя новый класс объектов некоторого вида, разработчик может изменить для него начальные значения предопределенных атрибутов, а также декларировать дополнительные атрибуты.

Методы

Совокупность методов (или функций) класса объектов определяет набор элементарных действий, которые можно совершать над объектами этого класса. Методы позволяют изменять состояние объекта либо получать информацию о нем.

Примеры методов:

имя_объекта.Show( ) — метод любого графического объекта, делает объект видимым.



имя_окна.WorkSpaceWidth( ) — возвращает ширину рабочего пространства окна.

Здесь имя функции употребляется вместе с именем объекта. В этом специфика подхода.
Для каждого вида объектов в PowerBuilder предопределен набор методов. Определяя новый класс объектов некоторого вида, разработчик может описать дополнительные методы для объектов этого класса.

События

Событие – действие, совершенное по отношению к объекту пользователем или другим объектом.

Примеры:
Событие clicked возбуждается в объекте, если пользователь нажал левую кнопку мыши, когда ее указатель находился на изображении объекта на экране.
Событие open возбуждается в объекте-окне, когда его открывают при помощи функции Open.
Заметим, что любое событие в объекте можно возбудить искусственно, при помощи метода TriggerEvent, например:

имя_кнопки.TriggerEvent(clicked).

Для каждого вида объектов в PowerBuilder предопределен набор событий с пустой программой обработки. Определяя новый класс объектов, программист описывает обработку событий в объектах этого класса в виде программ (скриптов) на языке PowerScript. Он может также расширить набор событий для объектов класса.

Если программа обработки какого-либо события не задана, то считается, что объекты данного класса не реагируют на это событие.

Класс объектов PowerBuilder – множество объектов, обладающих одинаковым наборами атрибутов, методов, событий.

Прикладная система, созданная в PowerBuilder, функционирует как совокупность объектов, которые под воздействием событий могут изменять свои свойства и свойства других объектов, создавать и уничтожать другие объекты.

Виды объектов

Классы объектов в PowerBuilder подразделяются на несколько видов, называемых далее виды объектов. Вот некоторые из них:

- приложение (Application) – головной объект прикладной системы, с которого начинается ее выполнение.
- окно (Window) – прямоугольная область экрана, которая может содержать меню, а также органы управления – различного рода кнопки, текстовые и графические элементы, окна с данными из БД и др.
- меню (Menu) – иерархические меню, принятые в системе Windows.
- командная кнопка (Command Button) – орган управления, который может принадлежать окну.

Два глобальных системных объекта:
- error – объект типа error, используемый при обработке ошибок.
- sqlca – объект типа transaction, используемый при взаимодействии с базой данных.

Создание прикладной системы в PowerBuilder

Создание прикладной системы в PowerBuilder сводится к описанию множества классов объектов, составляющих систему. Описание класса объектов в свою очередь сводится к описанию свойств (набора атрибутов и их начальных значений), набора методов (функций) и поведения (набора событий и программ, описывающих обработку событий).

Когда объект-приложение запускается на выполнение, то в нем возбуждается событие open. Если обработка этого события в приложении не описана, возникает ошибка выполнения. В противном случае это событие обрабатываться. Как правило, в программе обработки события open в приложении открывается (создается) головное окно прикладной системы, которое содержит такие объекты как меню, различного рода кнопки, строки ввода данных и т.п.

Для открытия окна используется функция open(*имя_окна*)

Для закрытия окна можно использовать функцию close(parent).

Часто требуется, чтобы приложение на протяжении сеанса работы оставалось подключенным к определенной базе данных. В этом случае в сценарий события открытия приложения (open) помещается оператор CONNECT, встроенный оператор SQL.

Пример:

```
SQLCA.DBMS = "ODBC"
SQLCA.AutoCommit = False
SQLCA.DBParm = "ConnectString='DSN=EAS Demo DB V4;UID=dba;PWD=sql'"
```

CONNECT;

Переменная SQLCA – глобальная переменная объекта транзакций, в которой хранится информация о подключении к базе данных. Объекты транзакций используются при всех видах взаимодействий между приложением Power Builder и базой данных.

В сценарии события закрытия приложения используется оператор DISCONNECT, отменяющий подключение к базе данных.

Обработка ошибок является важной частью любого приложения. При выполнении операторов SQL, PowerBuilder помещает соответствующий код об ошибке и другую информацию в объект транзакций (sqlca). Разработчик приложения должен проанализировать этот код и обработать ошибку (если таковая имеется).

Для объектов транзакций существует свойство SqlCode, которое возвращает код, указывающий на удачу или неудачу выполнения запросов SQL. При отсутствии ошибок его значение равно 0.

Следовательно, для обработки ошибки подключения к базе данных можно использовать следующий код:

```
If SQLCA.SqlCode<>0 Then
…
End If
```

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Запустить PowerBuilder.
2. Открыть приложение, созданное в лабораторной работе N 3.
3. Написать сценарии события clicked для объектов приложения, позволяющих открывать и закрывать окна приложения.
4. Написать сценарии событий открытия и закрытия приложения, позволяющие осуществлять подключение и отмену подключения к базе данных, созданной в лабораторной работе N 2.
5. Предусмотреть обработку ошибки подключения к базе данных. В случае невозможности подключения следует вывести на экран предупреждающее сообщение с помощью функции MessageBox.
6. Установить порядок перехода фокуса по элементам управления окон приложения с помощью пункта меню Format ->Tab Order.

4. ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать наименование работы, цель работы, задание на работу, порядок перехода фокуса по элементам управления окон приложения, сценарий события открытия приложения, тестовый пример – реакция на ошибку подключения к базе данных, вывод.

ЛАБОРАТОРНАЯ РАБОТА N 5
РАЗРАБОТКА ПРИЛОЖЕНИЙ В СРЕДЕ POWERBUILDER. РАБОТА С БАЗАМИ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение синтаксиса встроенных запросов SQL и приобретение практических навыков организации взаимодействия приложения PowerBuilder с базами данных.

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СПРАВКА

Существует два основных способа, с помощью которых приложения PowerBuilder взаимодействуют с базами данных. Первый – написание сценария на языке PowerScript, с использованием запросов SQL. Второй способ – это использование встроенного в Power Builder средства DataWindow. В данной лабораторной работе будет рассмотрен первый способ.

Все операции, которые приложение выполняет с базой данных, осуществляются с помощью запросов SQL. В зависимости от способа формирования, запросы SQL в PowerScript можно разделить на две категории:

- встроенные запросы SQL (Embedded SQL) – запросы, встраиваемые непосредственно в сценарий вместе с обычными операторами PowerScript.
- динамические запросы SQL (Dynamic SQL) – запросы SQL, которые записываются в строковых переменных с последующей передачей данных для выполнения.

Использование встроенных запросов SQL является более простым способом. Динамические запросы SQL предоставляют дополнительные возможности и гибки в применении. Так, с их помощью вводятся запросы, использующие те параметры, которые еще не были известны в начале работы приложения.

К встроенным запросам SQL относятся: CLOSE, DECLARE, EXECUTE, OPEN, SELECTBLOB, COMMIT, DELETE, FETCH, ROLLBACK, UPDATE, CONNECT, DISCONNECT, INSERT, SELECT, UPDATEBLOB.

Приведенный список запросов не исчерпывает всех возможностей использования SQL в PowerScript. Например, в качестве запроса SQL можно использовать оператор из PowerScript CREATE TABLE, хотя такой запрос отсутствует в списке стандартных запросов SQL.

Рассмотрим синтаксис некоторых запросов.

Следующий запрос добавляет в таблицу новую строку:

INSERT INTO *название_таблицы*
(*поле1*, *поле2*)
VALUES (‘*значение1*’,’*значение2*’);

Ниже приведен пример удаления из таблицы всех строк со значением *значение* в поле *поле*:

DELETE FROM *название_таблицы*
WHERE *поле* =’*значение*’;

Встроенный SQL-запрос SELECT позволяет считывать отдельную строку из базы данных:

String s1, s2

SELECT *поле1*, *поле2*
INTO :s1, :s2
FROM *название_таблицы*
WHERE *поле3=’значение3’* and *поле4*=’’;

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

1. Запустить PowerBuilder.
2. Открыть приложение, созданное в лабораторной работе N 3.
3. Реализовать типовые задачи:
- Ввод данных в базу;
- Поиск в базе данных, отвечающих заданным условиям.
4. Предусмотреть обработку возможных ошибок.

4. ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать наименование работы, цель работы, задание на работу, сценарии приложения, позволяющие реализовать ввод данных в базу и поиск в базе данных, отвечающих заданным условиям, тестовые примеры, вывод.

ЛАБОРАТОРНАЯ РАБОТА N 6
РАЗРАБОТКА ПРИЛОЖЕНИЙ В СРЕДЕ POWERBUILDER. СОЗДАНИЕ ОКНА ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Целью работы является приобретение практических навыков создания и использования в приложении окон данных.

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СПРАВКА

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:

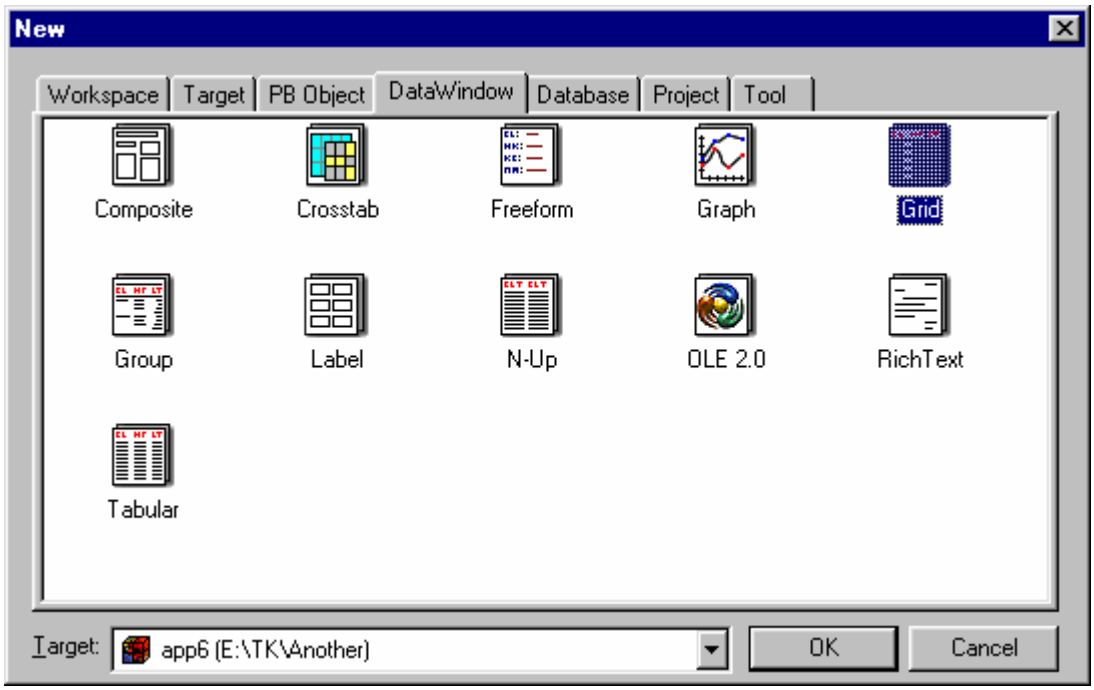
1. Объект DataWindow, создаваемый в мастерской окон данных (DataWindow Painter) и встраиваемый в элемент управления DataWindow в окне PowerBuilder.
2. Окно, разрабатываемое в мастерской окон (Window Painter). В нем располагаются все остальные объекты и элементы управления.
3. Элемент управления DataWindow, 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 DataWindow.
4. 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). Для вызова мастера окон данных нужно нажать кнопку New панели инструментов и выбрать закладку DataWindow.



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 в лабораторной работе N 2, 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.

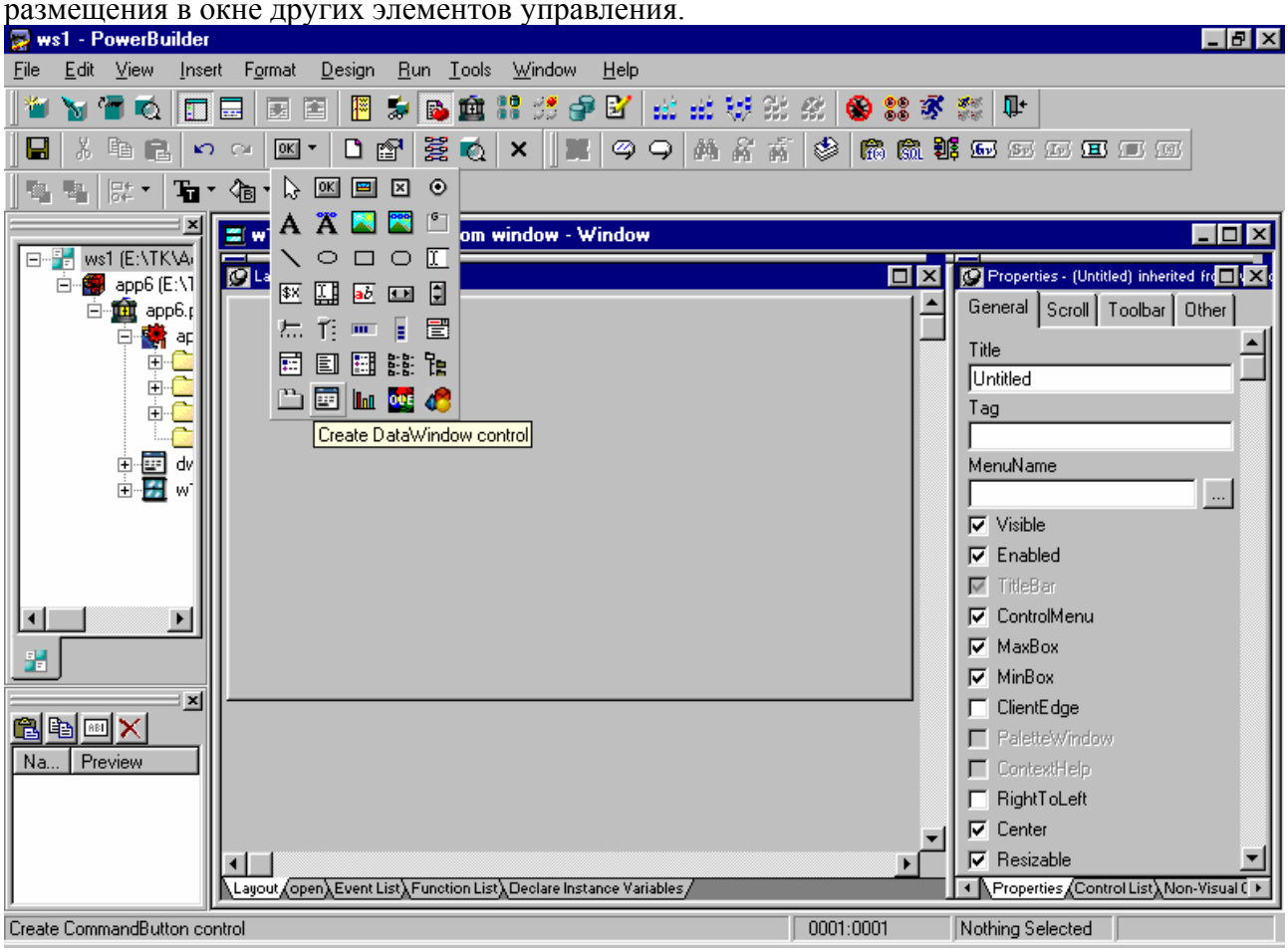
1. Quick Select (Быстрый выбор). Основываясь на сделанных разработчиком приложения настройках, PowerBuilder генерирует оператор SQL SELECT, который можно редактировать в мастере Select Painter.
2. SQL Select (Выбор SQL-запроса). Выбор пиктограммы SQL Select приведет непосредственно к вызову мастера Select 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.

Поскольку вы выбрали среди источников данных опцию Quick Select, 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 элемента управления DataWindow практически ничем не отличается от размещения в окне других элементов управления.



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

Обратим внимание, что объект управления DataWindow и элемент управления 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.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Запустить PowerBuilder.
2. Открыть рабочее приложение.
3. Создать в одном из окон приложения окно данных DataWindow, руководствуясь описанием последовательности действий, приведенном в краткой теоретической справке.
4. Используя DataWindow, реализовать в приложении решение типовой задачи:

- Просмотр и редактирование данных

Отчет по лабораторной работе должен содержать наименование работы, цель работы, задание на работу, сценарии приложения, позволяющие реализовать просмотр и редактирование данных, тестовые примеры, вывод.

Лабораторная работа №7

ВВЕДЕНИЕ В СИСТЕМУ PROTEGE.

Цель работы

Целью работы является изучение понятия «онтология» и свойств онтологии, а также знакомство с системой Protégé.

1. Краткая теоретическая справка

Что такое онтология?

Онтология описывает основные концепции (положения) предметной области и определяет отношения между ними.
Процесс построения онтологий состоит из создания следующих блоков:

· Классов и их свойств (classes, properties).

· Свойств каждой концепции, описывающих различные функциональные возможности и атрибуты концепции (слоты (slots), иногда называемые роли).

· Ограничения по слотам (также известных как аспекты/грани (slot facets), иногда называемые ограничения ролей).

Онтология вместе с множеством индивидуальных экземпляров классов составляют базу знаний.

В литературе по искусственному интеллекту содержится много определений понятия онтологии, многие из которых противоречат друг другу. В этой статье **онтология** – формальное явное описание понятий в рассматриваемой предметной области (**классов,** иногда их называют **понятиями**), свойств каждого понятия, описывающих различные свойства и атрибуты понятия (**слотов** (иногда их называют **ролями** или **свойствами**)), и ограничений, наложенных на слоты (**фацетов,** иногда их называют **ограничениями ролей**). Онтология вместе с набором индивидуальных **экземпляров** классов образует **базу знаний**. В действительности, трудно определить, где кончается онтология и где начинается база знаний.

Зачем создавать онтологию?

В последние годы разработка онтологий – явное формальное описание терминов предметной области и отношений между ними – переходит из мира лабораторий по искусственному интеллекту на рабочие столы экспертов по предметным областям. Во всемирной паутине онтологии стали обычным явлением. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты (как на сайте Yahoo!), до категоризаций продаваемых товаров и их характеристик (как на сайте Amazon.com). Во многих дисциплинах сейчас разрабатываются стандартные онтологии,

которые могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в своей области. Например, в области медицины созданы большие стандартные, структурированные словари, такие как SNOMED и семантическая сеть Системы Унифицированного Медицинского Языка (the Unified Medical Language System). Также появляются обширные общецелевые онтологии. Например, Программа ООН по развитию (the United Nations Development Program) и компания Dun & Bradstreet объединили усилия для разработки онтологии UNSPSC, которая предоставляет терминологию товаров и услуг (http://www.unspsc.org/).

Онтология определяет общий словарь для ученых, которым нужно совместно использовать информацию в предметной области. Она включает машинно-интерпретируемые формулировки основных понятий предметной области и отношения между ними.

Почему возникает потребность в разработке онтологии? Вот некоторые причины:

· Для совместного использования людьми или программными агентами общего понимания структуры информации.
· Для возможности повторного использования знаний в предметной области.
· Для того чтобы сделать допущения в предметной области явными.
· Для отделения знаний в предметной области от оперативных знаний.
· Для анализа знаний в предметной области.

Совместное использование людьми или программными агентами общего понимания структуры информации является одной из наиболее общих целей разработки онтологий. К примеру, пусть несколько различных веб-сайтов содержат информацию по медицине или предоставляют информацию о платных медицинских услугах, оплачиваемых через Интернет. Если эти веб-сайты совместно используют и публикуют одну и ту же базовую онтологию терминов, которыми они все пользуются, то компьютерные агенты могут извлекать информацию из этих различных сайтов и накапливать ее. Агенты могут использовать накопленную информацию для ответов на запросы пользователей или как входные данные для других приложений.

Обеспечение возможности использования знаний предметной области стало одной из движущих сил недавнего всплеска в изучении онтологий. Например, для моделей многих различных предметных областей необходимо сформулировать понятие времени. Это представление включает понятие временных интервалов, моментов времени, относительных мер времени и т.д. Если одна группа ученых детально разработает такую онтологию, то другие могут просто повторно использовать ее в своих предметных областях. Кроме того, если нам нужно создать большую онтологию, мы можем интегрировать

несколько существующих онтологий, описывающих части большой предметной области. Мы также можем повторно использовать основную онтологию, такую как UNSPSC, и расширить ее для описания интересующей нас предметной области.

Создание явных допущений в предметной области, лежащих в основе реализации, дает возможность легко изменить эти допущения при изменении наших знаний о предметной области. Жесткое кодирование предположений о мире на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно изменить, особенно непрограммисту. Кроме того, явные спецификации знаний в предметной области полезны для новых пользователей, которые должны узнать значения терминов предметной области.

Отделение знаний предметной области от оперативных знаний – это еще один вариант общего применения онтологий. Мы можем описать задачу конфигурирования продукта из его компонентов в соответствии с требуемой спецификацией и внедрить программу, которая делает эту конфигурацию независимой от продукта и самих компонентов. После этого мы можем разработать онтологию компонентов и характеристик ЭВМ и применить этот алгоритм для конфигурирования нестандартных ЭВМ. Мы также можем использовать тот же алгоритм для конфигурирования лифтов, если мы предоставим ему онтологию компонентов лифта.

Анализ знаний в предметной области возможен, когда имеется декларативная спецификация терминов. Формальный анализ терминов чрезвычайно ценен как при попытке повторного использования существующих онтологий, так и при их расширении.

Часто онтология предметной области сама по себе не является целью. Разработка онтологии сродни определению набора данных и их структуры для использования другими программами. Методы решения задач, доменно-независимые приложения и программные агенты используют в качестве данных онтологии и базы знаний, построенные на основе этих онтологий.

Как создать онтологию?

Строго говоря, единого универсального подхода к созданию онтологий, который бы привел к однозначно успешному результату не существует. Процесс создания онтологий обычно является итеративным, т.е. сначала создается черновой набросок, а затем по мере необходимости происходит возврат для определения деталей, и так продолжается до тех пор, пока онтология не будет отражать концепцию предметной области с определенной степенью.

Практически, создание онтологий включает:

1. Определение классов в онтологии,
2. Организация классов в некоторую иерархию (базовый класс –> подкласс),
3. Определение слотов и их допустимых значений,
4. Заполнение значений слотов для экземпляров классов.

Как определить, правильно ли создана онтология?

Для любой предметной области может существовать бесчисленное количество онтологий; ведь каждая новая онтология – это всего лишь еще один из способов структурирования концепций и отношений между ними. Однако существуют несколько простых принципов, которые могут помочь при принятии решений о том, как создавать те или иные онтологии:

· Не может быть только одного способа описания модели предметной области – всегда есть жизнеспособная альтернатива. Лучшее решение почти всегда будет зависеть от того, какая система разрабатывается и от возможных будущих изменений в системе.

· Процесс разработки обязательно должен быть итеративным.

· Концепции в онтологии должны быть максимально близки к объектам (логическим или физическим) и отношениям между ними в интересуемой области знаний. При правильном моделировании, онтология может быть представлена предложениями, где вероятней всего в качестве существительных будут объекты, а отношений – глаголы.

С чего начать?

Для начала необходимо понять, для чего должна быть использована онтология и как примерно выглядел бы ее детальный и общий вариант. Затем среди многих альтернатив вы должны будете выбрать ту, которая будет лучше всего работать для намеченной задачи, а также будет наиболее интуитивна, расширяема, поддерживаема. При этом необходимо помнить, что онтология представляет предметную область в реальном окружающем мире, и потому понятия в онтологии также должны отражать эту реальность. После того как вы сделаете черновой вариант онтологии, вы можете проверить ее и откорректировать, используя Protégé, или путем обсуждения с экспертами в исследуемой области (как результат, почти наверняка придется пересмотреть черновой вариант). Такой процесс итеративной коррекции, вероятней всего, будет продолжаться на протяжении всего жизненного цикла онтологии.

Разработка простейшей системы

Основные положения

Предположим, что мы хотим разработать систему, которая помогает управлять стоимостью и организацией печатного издания (для простоты можно взять некую газету). Система должна отвечать на следующие вопросы:

· Кто ответственный за каждый раздел в газете?

· Каково содержимое каждой статьи в разделе и кто автор?

· Перед кем отчитывается каждый автор?

· Каково расположение и расходы на каждую статью?

После того как мы определились с идеей, мы можем расписать некоторые из важных положений системы. Сюда могут войти: основные концепции и их свойства, а также отношения между ними. Для начала мы можем просто определить термины, независимо от роли, которую они могут играть в онтологии.

Итак, в любой газете есть разделы. Каждый раздел имеет содержимое, например, статьи, реклама и т.д. и ответственного редактора. У каждой статьи есть автор, который может быть как работником газеты, так и быть приглашенным со стороны. Для каждого автора, работающего в газете, мы хотим знать его имя и зарплату, а также перед кем он отчитывается.
По мере определения понятий, мы неявно определяем рамки нашей онтологии, а именно, что мы должны будем включить в нашу модель, а что нет. К примеру, при начальном рассмотрении термина «работник», мы, возможно, хотели бы включить в это понятие вахтера или водителя грузовика из службы доставки. Однако, подумав, мы могли осознать, что хотим чтобы наша онтология была сфокусирована на производственных затратах, связанных напрямую с тем что, как и где написано в газете. Таким образом, мы решаем не включать вахтера и т.п. в область рассмотрения.
Получив достаточно полный список терминов, мы можем разделить эти понятия по категориям в зависимости от их функции в онтологии. Понятия (концепции/термины предметной области), являющиеся объектами, такие как статья или автор, будут представлены в виде классов. Свойства классов, такие как содержимое раздела или зарплата, могут быть представлены как слоты, а ограничения на свойства или отношения между классами как грани/аспекты (slot facets).
Определив основные понятия, теперь мы можем показать, как создавать и структурировать их, используя систему Protégé.

Порядок выполнения работы.

Создание проекта

Перед началом работы, вы должны создать новый проект в системе Protégé. Для этого:
1. Запустите Protégé. Если у вас уже открыт проект, просто сохранитесь и перезапустите программу. После того как программа запустилась, появляется диалог приветствия, предлагающий создать новый проект, открыть последний проект или посмотреть документацию.

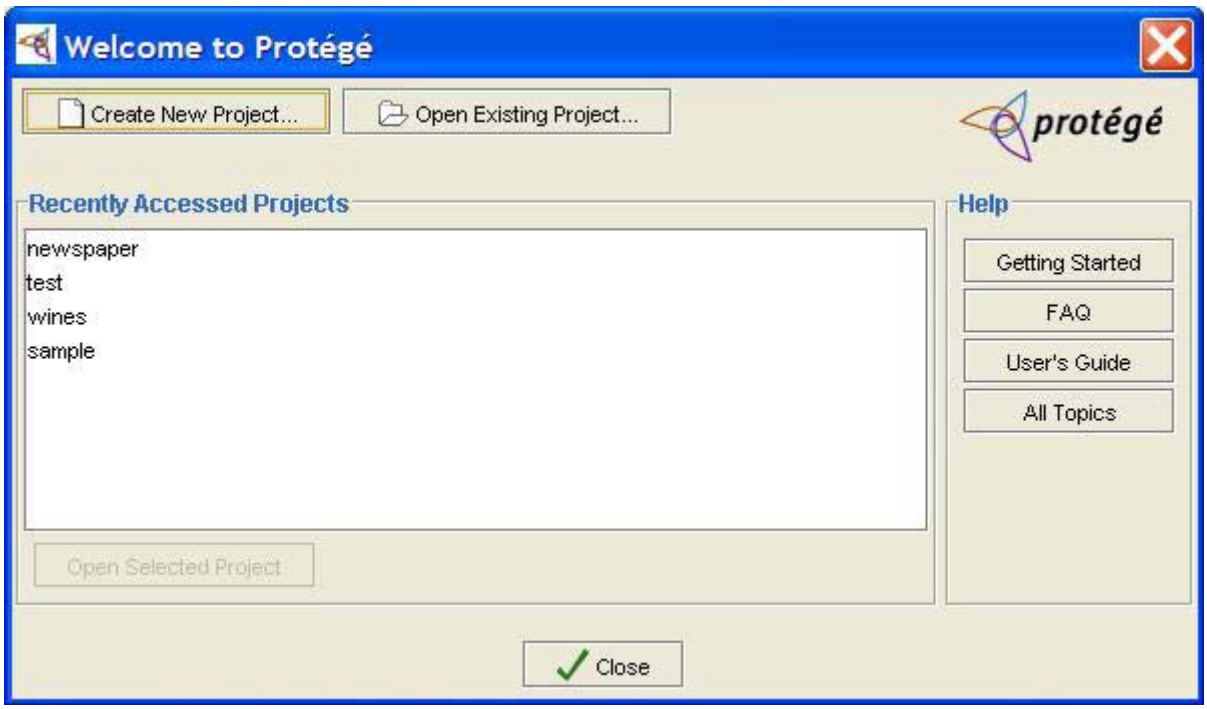


Рисунок 1. Окно приветствия.

2. Щелкните мышкой по кнопке **Create New Project...**. Появится диалоговое окно "Create New Project", позволяющее выбрать тип проекта. Если у вас нет необходимости в специальном формате для ваших файлов, просто нажмите кнопку **Finish** – будет выбран формат файла по умолчанию Protege Files (.pont and pins).

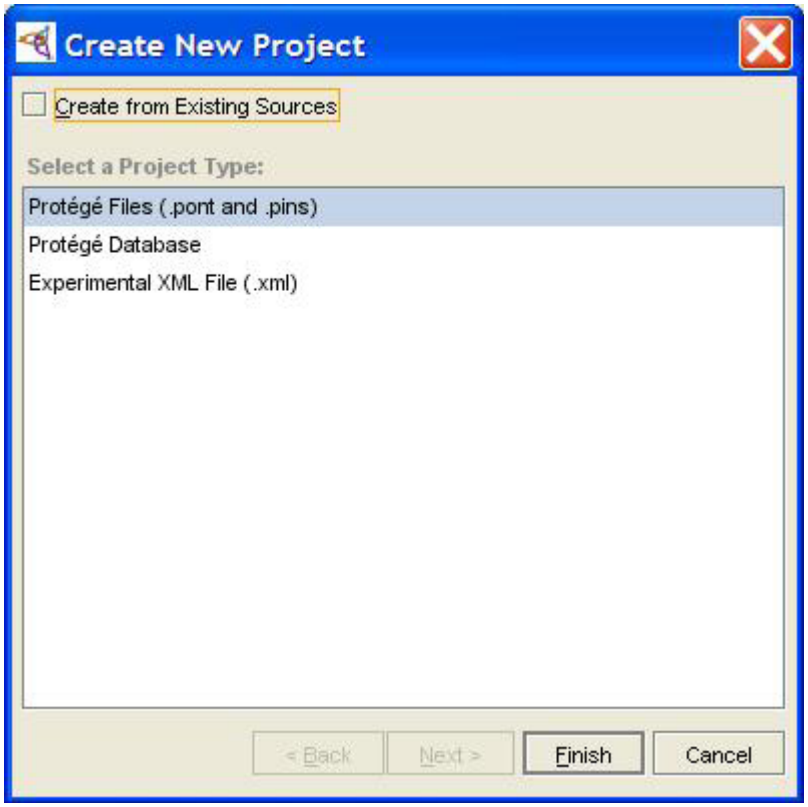


Рисунок 2. Окно создания нового проекта.

3. Откроется окно проекта Protégé. Новый проект всегда открывается в

области просмотра классов (Classes view). Видно, что в этой области на данный момент находятся только внутренние системные класс Protégé: THING и SYSTEM-CLASS. Никаких экземпляров классов создано к этому моменту не будет.

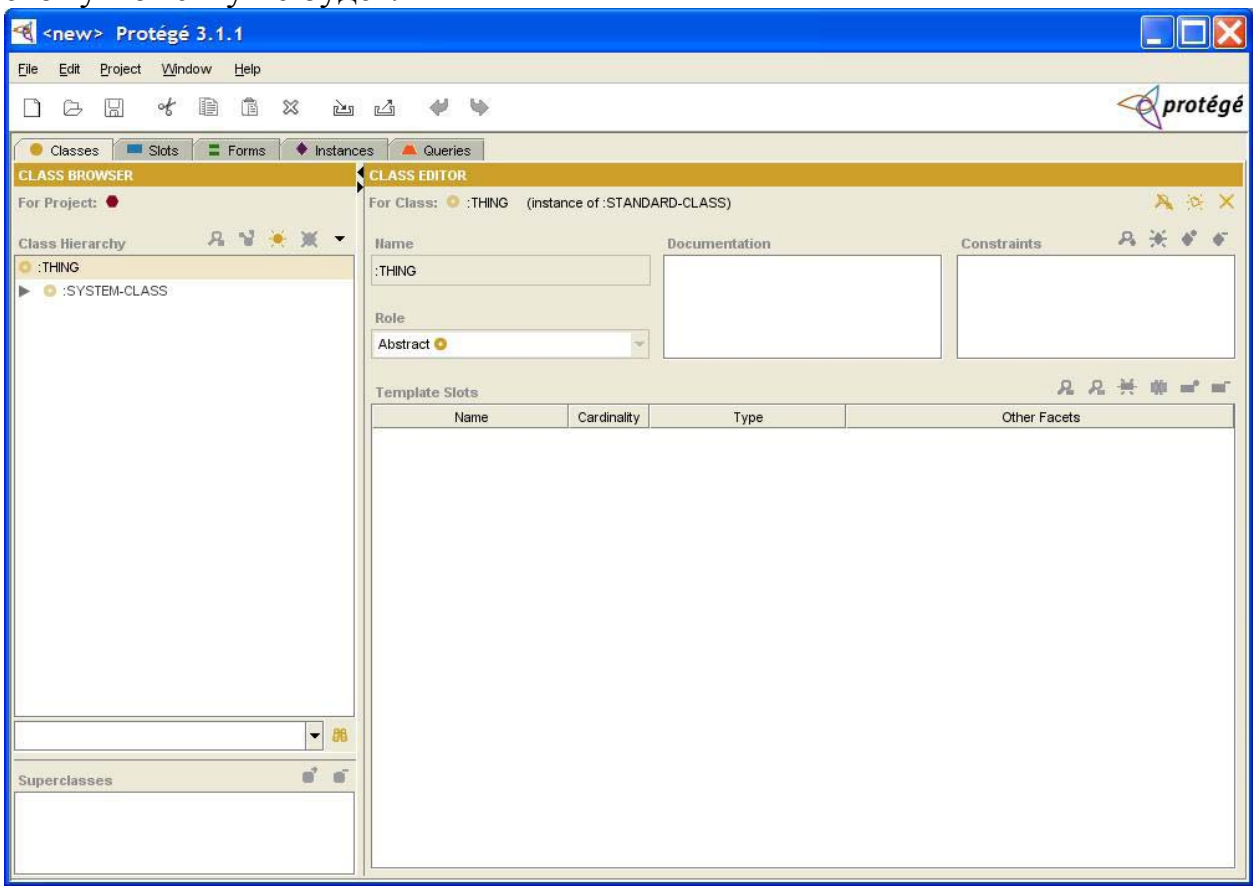


Рисунок 3. Область просмотра классов

Сохранение проекта

Во время работы с программой вы можете захотеть сохранить промежуточные изменения, для этого:

1. Щелкните кнопку «сохранить проект», вы также можете выбрать пункт **Save project** из меню файл **File**.

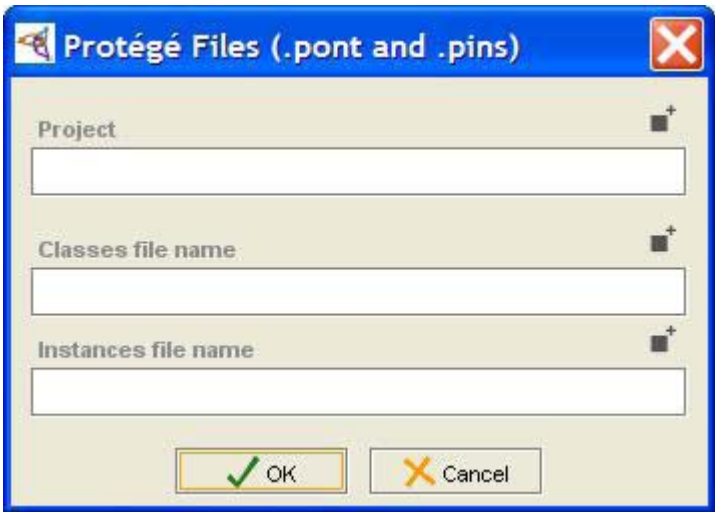


Рисунок 4. Окно сохранения проекта.

2. Для того чтобы указать место, куда будет сохранен Ваш проект, нажмите кнопку чуть выше самой верхней строчки (напротив надписи Project). В открывшемся диалоговом окне, перейдите по нужному пути в файловой системе (или создайте каталог, где будут храниться данные проекта и откройте созданную папку).

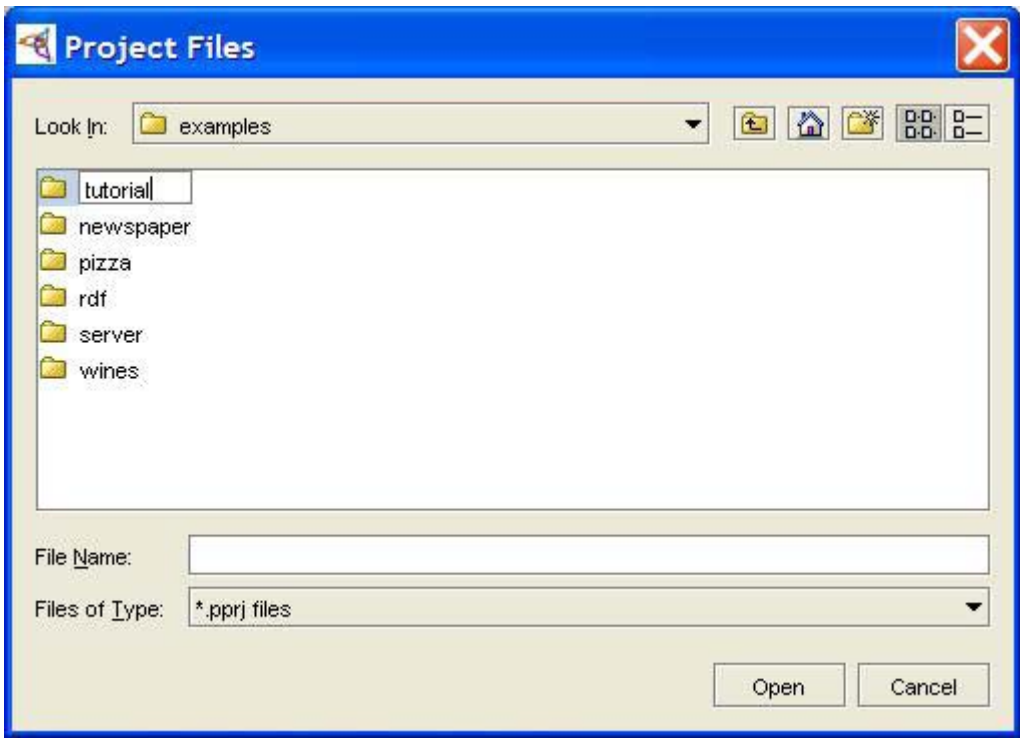


Рисунок 5. Окно выбора имени файла проекта.

3. Введите имя файла проекта (например, "tutorial").
4. Нажмите кнопку **Select**.
5. Вы вернетесь в окно сохранения файлов проекта, нажмите **OK** и проект будет сохранен.

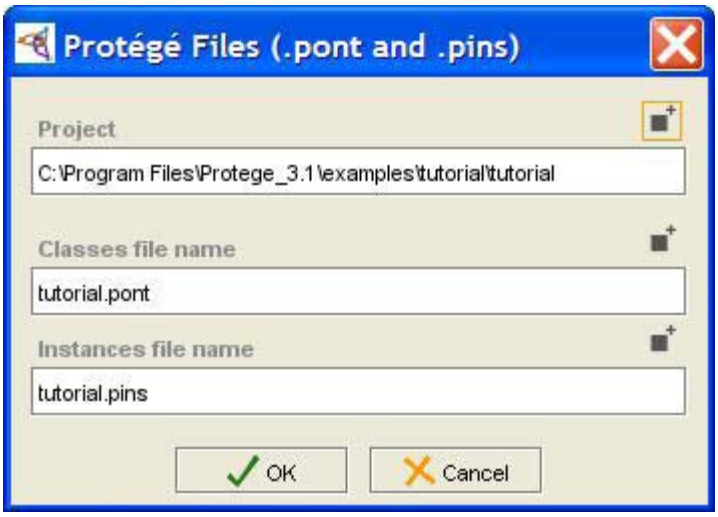


Рисунок 6. Завершение сохранение проекта

2. Порядок выполнения работы

- Получить задание – определенную предметную область.
- Запустить приложение
- Создать новую онтологию, кратко описать ее в разделе Documentation и сохранить.

ЛАБОРАТОРНАЯ РАБОТА № 8

Создание онтологий в системе Protege

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение принципов построения онтологий и приобретение первоначальных практических навыков работы с системой Protege.

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СПРАВКА

Онтология описывает основные концепции (положения) предметной области и определяет отношения между ними.

Процесс построения онтологий состоит из создания следующих блоков:

· Классов и их свойств (classes, properties).

· Свойств каждой концепции, описывающих различные функциональные возможности и атрибуты концепции (слоты (slots), иногда называемые роли).

· Ограничения по слотам (также известных как аспекты/грани (slot facets), иногда называемые ограничения ролей).

В литературе по искусственному интеллекту содержится много определений понятия онтологии, многие из которых противоречат друг другу. Будем использовать следующее определение: **онтология** – формальное явное описание понятий в рассматриваемой предметной области (**классов,** иногда их называют **понятиями**), свойств каждого понятия, описывающих различные свойства и атрибуты понятия (**слотов** (иногда их называют **ролями** или **свойствами**)), и ограничений, наложенных на слоты (**фацетов,** иногда их называют **ограничениями ролей**). Онтология вместе с набором индивидуальных **экземпляров** классов образует **базу знаний**. В действительности, трудно определить, где кончается онтология и где начинается база знаний.

Редактор Protege

С момента его создания Protege многие годы использовался экспертами в основном для концептуального моделирования в области медицины. В последнее время его стали применять в других предметных областях — в частности, при создании онтологий для Semantic Web.

Используемые формализмы и форматы

Изначально единственной моделью знаний, поддерживаемой Protege, была фреймовая модель. Этот формализм сейчас является "родным" для редактора, но не единственным.

Protege имеет открытую, легко расширяемую архитектуру и помимо фреймов поддерживает все наиболее распространенные языки представления знаний

(SHOE, XOL, DAML+OIL, RDF/RDFS, OWL). Protege поддерживает модули расширения функциональности (plug-in). Расширять Protege для использования нового языка проще, чем создавать редактор этого языка "с нуля".
Protege основан на модели представления знаний OKBC (Open Knowledge Base Connectivity). Основными элементами являются классы, экземпляры, слоты (представляющие свойства классов и экземпляров) и фасеты (задающие дополнительную информацию о слотах).

Пользовательский интерфейс
Пользовательский интерфейс состоит из главного меню и нескольких вкладок для редактирования различных частей базы знаний и ее структуры. Набор и названия вкладок зависят от типа проекта (языка представления) и могут быть настроены вручную. Обычно имеются следующие основные вкладки: Classes, Slots, Forms, Instances, Queries.
Назначение основных вкладок — предоставить набор форм для заполнения базы знаний.

3. Создание проекта

Перед началом работы, нужно создать новый проект в системе Protégé. Для этого:
1. Запустить Protégé. Если уже открыт проект, надо просто сохраниться и перезапустить программу. После того как программа запустилась, появляется диалог приветствия, предлагающий создать новый проект, открыть последний проект или посмотреть документацию.

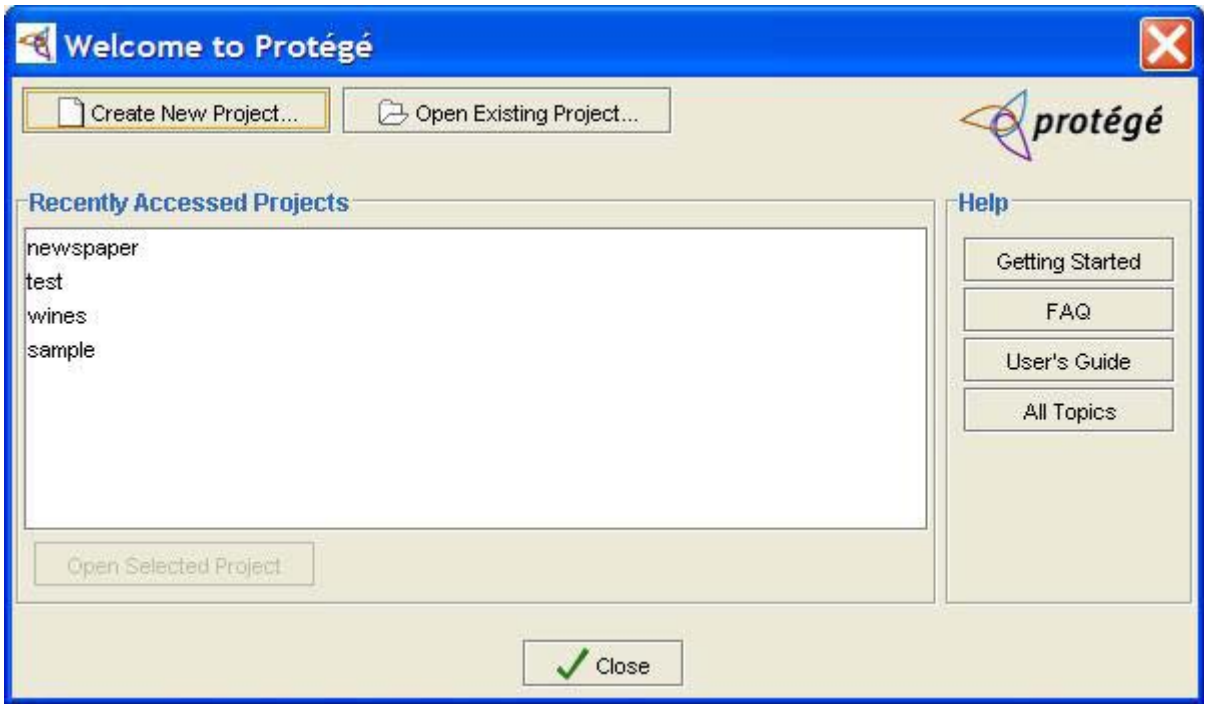


Рисунок 1. Окно приветствия.

2. Щелкнуть мышкой по кнопке **Create New Project...**. Появится диалоговое окно "Create New Project", позволяющее выбрать тип проекта. Если нет необходимости в специальном формате для файлов, просто нажать кнопку **Finish** – будет выбран формат файла по умолчанию Protege Files (.pont and pins).

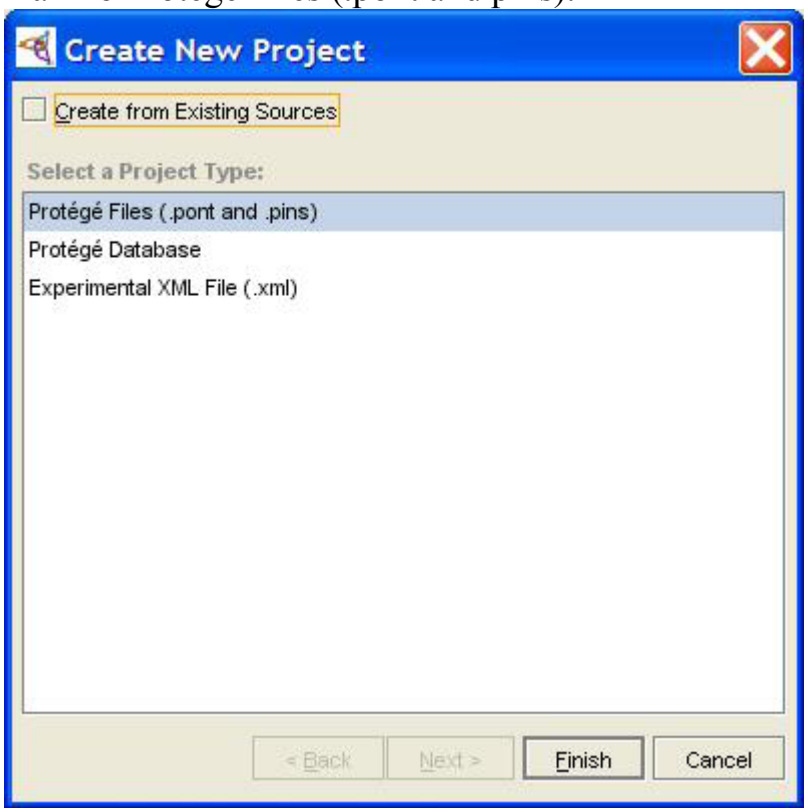


Рисунок 2. Окно создания нового проекта.

3. Откроется окно проекта Protégé. Новый проект всегда открывается в области просмотра классов (Classes view). Видно, что в этой области на данный момент находятся только внутренние системные класс Protégé: THING и SYSTEM-CLASS. Никаких экземпляров классов создано к этому моменту не будет.

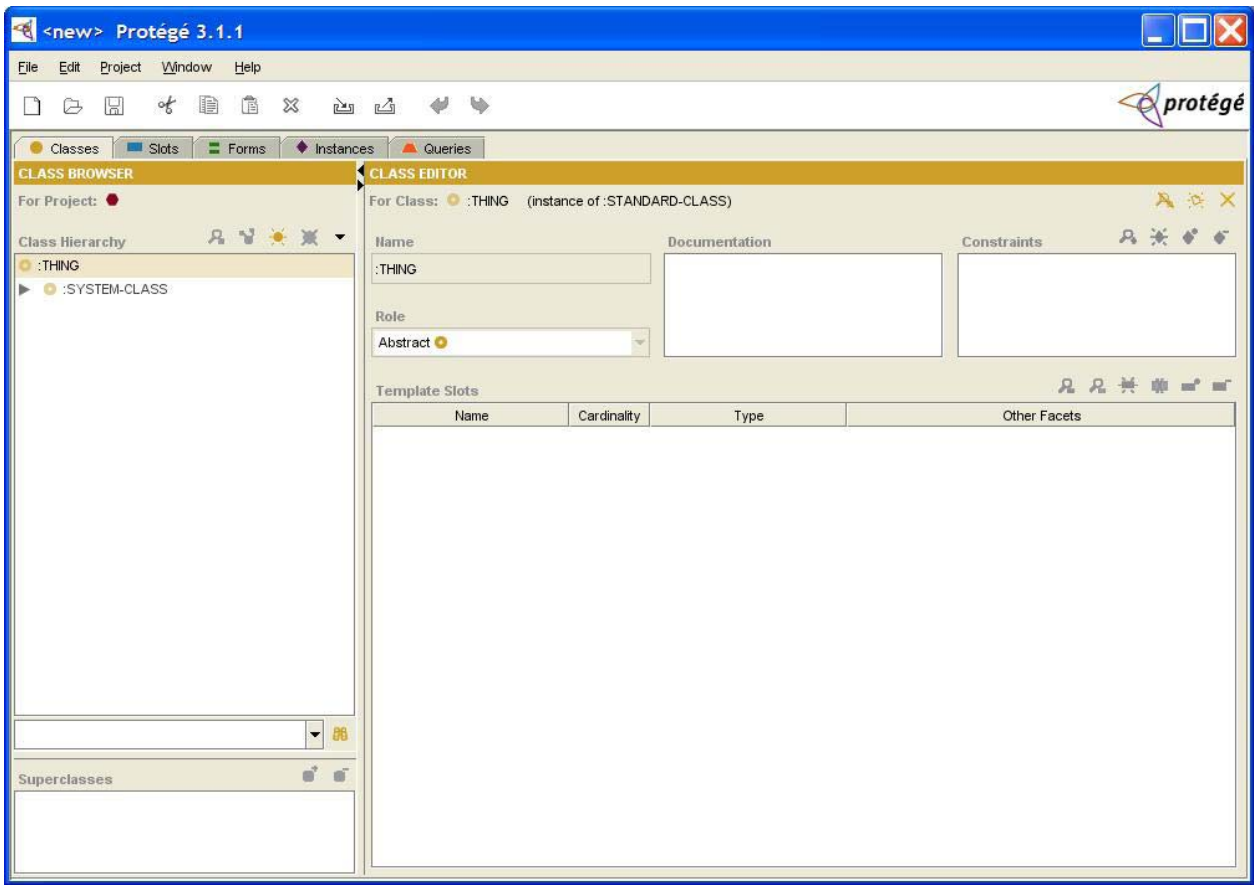


Рисунок 3. Область просмотра классов.

Сохранение проекта

Во время работы с программой можно сохранить промежуточные изменения, для этого:

1. Щелкнуть кнопку сохранить проект, также можно выбрать пункт **Save project** из меню файл **File**.

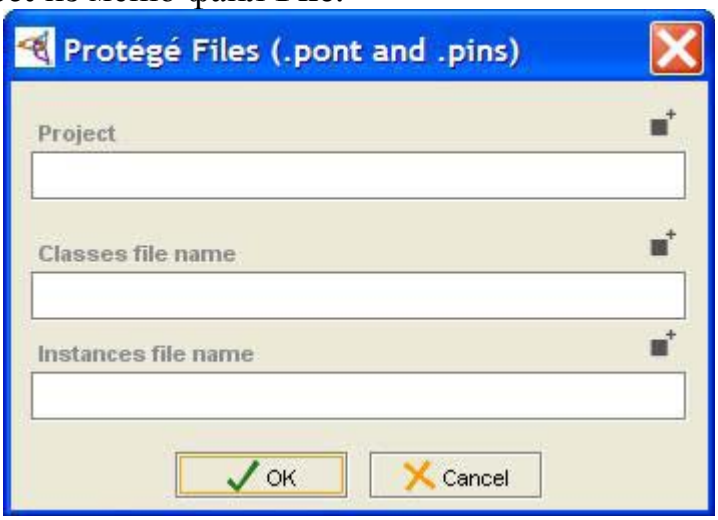


Рисунок 4. Окно сохранения проекта.

2. Для того чтобы указать место, куда будет сохранен проект, нажать кнопку чуть выше самой верхней строчки (напротив надписи Project). В открывшемся диалоговом окне, перейти по

нужному пути в файловой системе (или создать каталог, где будут храниться данные проекта и открыть созданную папку).

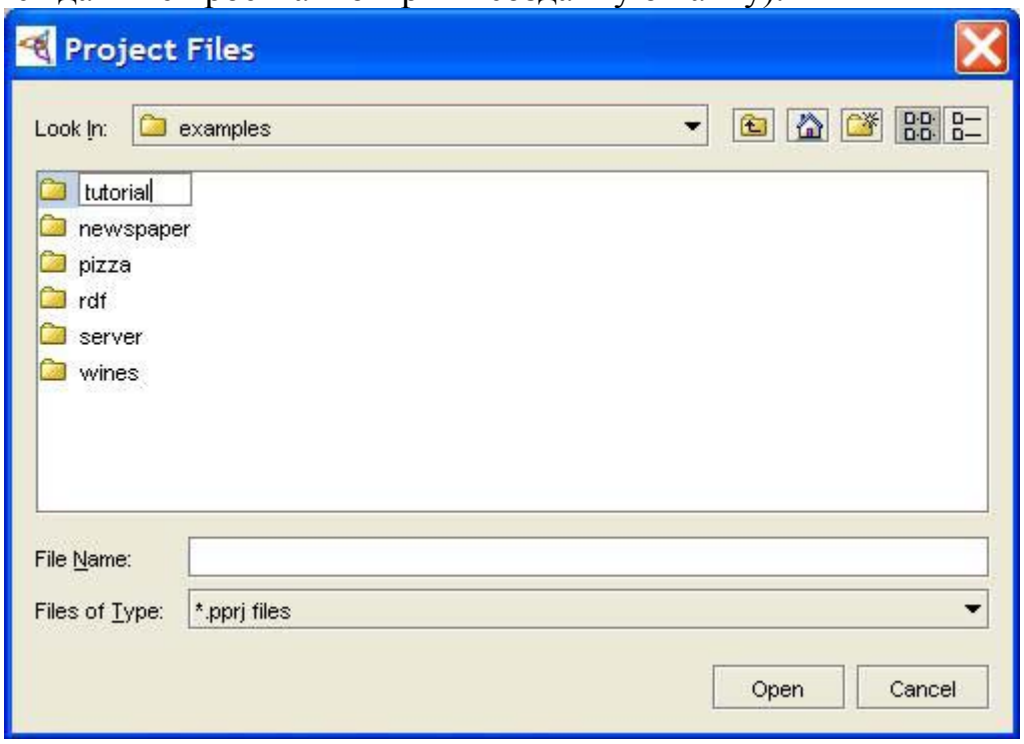


Рисунок 5. Окно выбора имени файла проекта.

4. Ввести имя файла проекта (например, "tutorial").
5. Нажать кнопку **Select**.
6. Произойдет возврат в окно сохранения файлов проекта, нажать **OK** и проект будет сохранен.

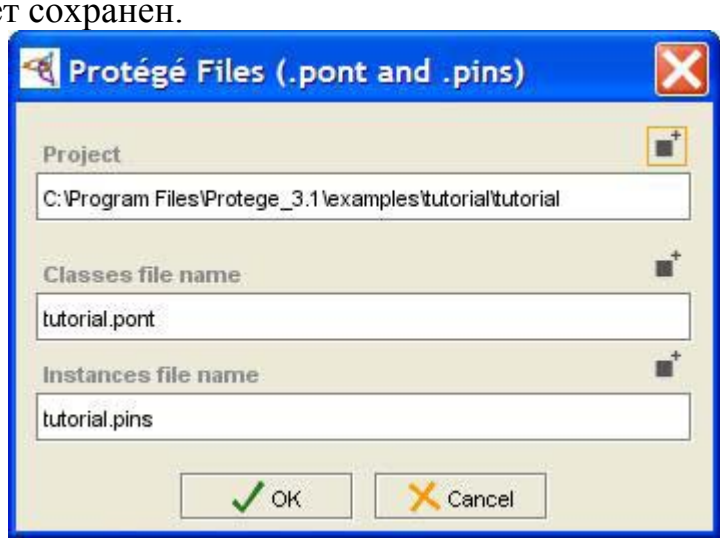


Рисунок 6. Завершение сохранение проекта.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

- Запустить Protege
- Создать новый проект, в соответствии со своей темой онтологии
- Сохранить проект

ЛАБОРАТОРНАЯ РАБОТА № 9

СОЗДАНИЕ КЛАССОВ ОНТОЛОГИЙ

1. ЦЕЛЬ РАБОТЫ

Целью работы является приобретение практических навыков работы с системой Protege

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СПРАВКА

Основное окно программы Protégé состоит из закладок (tabs) которые отображают различные аспекты модели знаний. Наиболее важной закладкой, в самом начале создания проекта, является закладка классов (Classes). Обычно классы соответствуют объектам или типам объектов в некой предметной области.

Классы в Protégé отображаются в виде иерархии наследования (inheritance hierarchy), которая располагается в области просмотра называемой Class Browser (или навигатор классов) в левой части закладки классов. Свойства классов выбранных в текущий момент в навигаторе, отображаются в редакторе классов справа.

Рассмотрим пример создания онтологии «Эволюционные вычисления». Сделаем в ней классы «Авторы», «Организации» и собственно «Эволюционные вычисления». В первые два класса будет заноситься информация о публикациях по данной тематике, в третий – описание терминов, относящихся к ЭВ.

СОЗДАНИЕ КЛАССА «АВТОРЫ»

1. Выбрать закладку классов.
2. Найти область в навигаторе классов (Class Browser), где отображается иерархия классов (Class Hierarchy, в окне Protégé слева). Эта область отображает иерархию классов, с выделенным текущим выбранным классом.

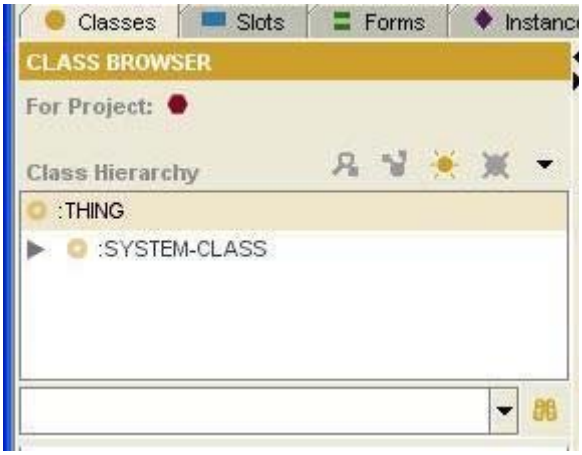


Рисунок 7. Область отображения иерархии классов.

3. Проверить: по умолчанию класс :THING (вещь, нечто) должен быть выделен.

Другой класс :SYSTEM_CLASS
используется для определения структур различных форм Protégé.
4. Нажать кнопку создать класс (Create Class) в верхнем правом углу навигатора классов. Новый класс будет создан со стандартным именем (основанном на имени проекта). В нашем случае “tutorial_Class_0”. Можно увидеть, что имя нового класса в навигаторе классов после создания будет выделено, для указания того, что этот класс выбран в данный момент.

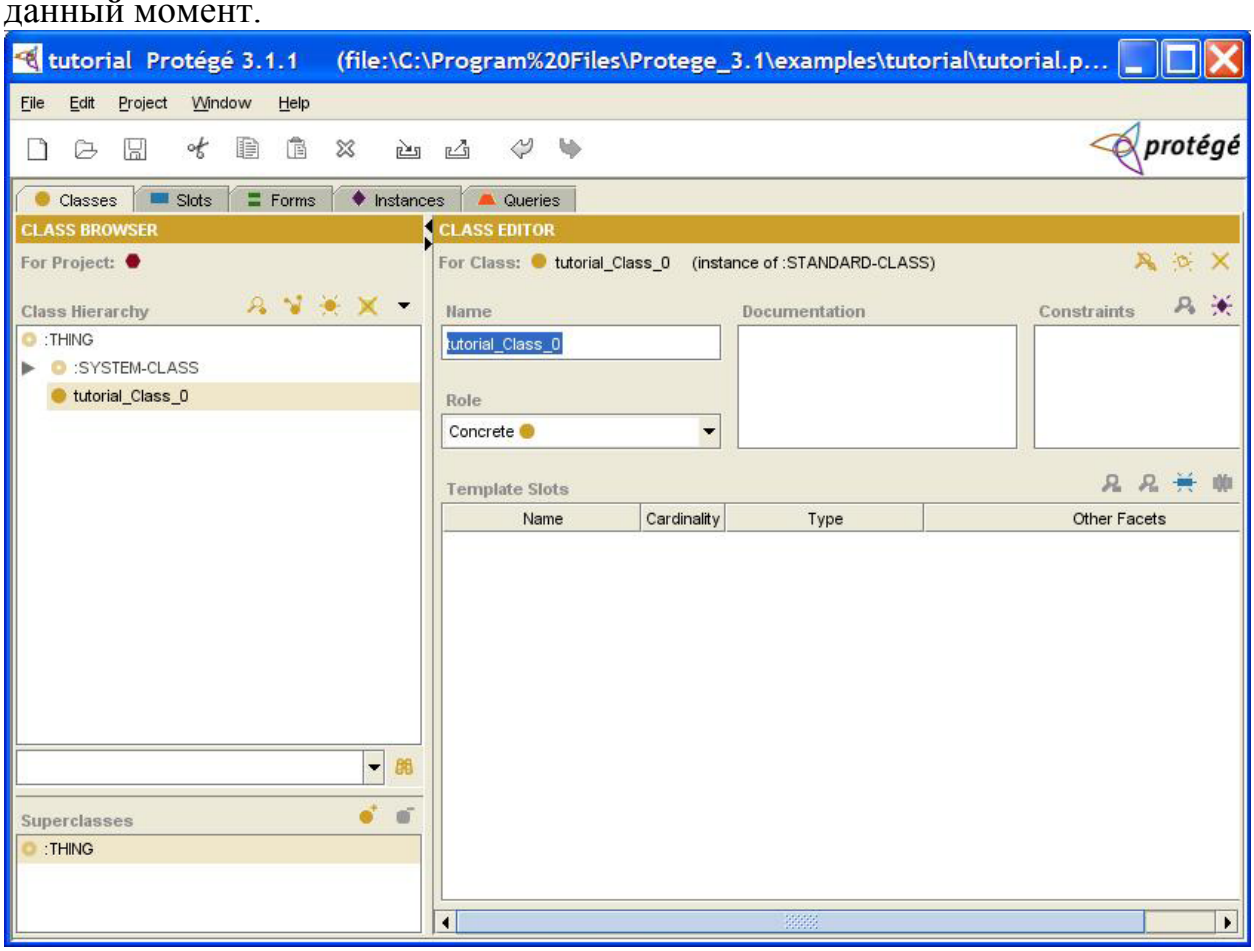


Рисунок 8. Редактор класса.

5. В активном поле редактора классов ввести “Авторы”. В системе Protégé приняты правила наименования, когда первая буква в каждом слове в имени класса пишется в верхнем регистре, а остальные буквы в нижнем, при этом слова разделяются символом подчеркивания.
6. Нажать ввод или щелкнуть мышью на отображаемый класс, чтобы подтвердить и отобразить свои изменения.
7. Если при изменении имени класса возникли проблемы, посмотреть в панель редактора классов справа в главном окне Protégé. Стандартное имя нового класса должно быть отображено и выделено в поле Name. Если правильное стандартное имя отображается, но не выделено, просто щелкнуть на поле **Name** мышкой, для того чтобы его отредактировать. Если имя неправильное, тогда скорей всего выбран неверный класс в области отображения иерархии классов