Практическая работа 47
Проектирование базы данных с использованием CASE-средств

Цель занятия: Получить практический опыт проектирования базы данных с использованием CASE-средств.

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio, Access)
DBDesigner online

Краткие теоретические сведения

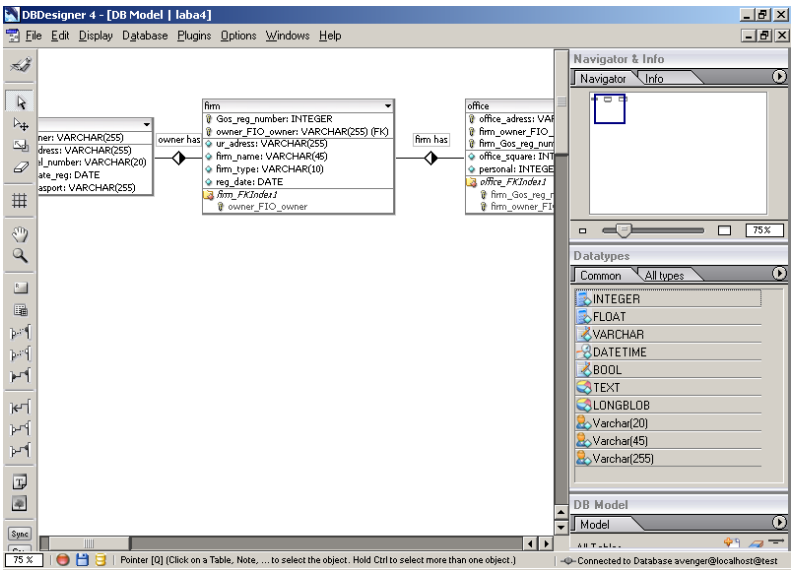
CASE-средства представляют собой программные средства, поддерживающие процессы создания и/или сопровождения информационных систем, такие как: анализ и формулировка требований, проектирование баз данных и приложений, генерация кода, тестирование, обеспечение качества, управление конфигурацией и проектом.

CASE-систему можно определить как набор CASE-средств, имеющих определенное функциональное предназначение и выполненных в рамках единого продукта.

DBDesigner – это свободно распространяемая CASE-система, предназначенная для проектирования, моделирования, создания и поддержки информационных систем. Программа может использоваться для Windows 2000/XP, LinuxKDE/Gnome и MySQL. DBDesigner позволяет:

- создавать модель проектируемой системы;
- преобразовывать модели системы в SQL-код, который можно использовать для создания базы данных с помощью DBDesigner или другого средства;
- проводить реинжиниринг – построение исходной модели программной системы путем исследования ее программных кодов. Эта функция очень удобна в случае, если необходимо разобраться уже существующей базе данных. Для проведения реинжиниринга следует выбрать в меню Database – ReversEngineering;
- создавать базу данных и автоматически вносить в нее изменения, используя соединение с сервером и синхронизацию;
- создавать SQL-запросы для внесения изменений и проведения операций над данными.

Пользовательский интерфейс программы:



Основные этапы проектирования базы данных:

1. Описание предметной области. Определение цели создания базы данных.

2. Определение сущностей предметной области (таблиц), которые должна содержать база данных.

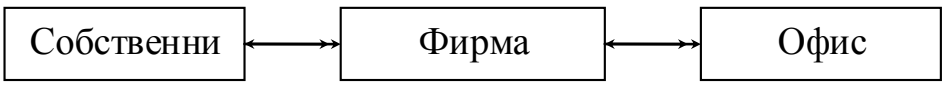
3. Определение атрибутов сущностей (необходимых в таблицах полей).

4. Построение инфологической модели. Определение связей между сущностями (таблицами).

Предметная область:

База данных содержит информацию о фирмах зарегистрированных в России.

Инфологическая модель:



Собственник (**ФИО собственника**, Адрес проживания собственника, Номер телефона собственника);

Фирма (**Государственный регистрационный № фирмы**, Юридический адрес фирмы, Название фирмы, Тип фирмы);

Офис (**Адрес офиса**, Площадь офиса, Количество персонала офиса);

Датологическая модель:

Здесь составим саму реляционную модель проектируемой БД: раскроем все связи между сущностями как связи между ключами. Для этого добавим в одну из связываемых сущностей дополнительный атрибут – первичный ключ из другой сущности. Причем добавление будем осуществлять в ту из этих сущностей, где не нарушится понятие ее первичного ключа.

Собственник	Фирма	Офис
ФИО собственника	**Гос. рег. № фирмы**	**Адрес офиса**
Адрес проживания собственника	Юридический адрес фирмы	Площадь офиса
Номер телефона собственника	Название фирмы	Количество персонала офиса
	Тип фирмы	*Гос. рег. № фирмы*
	ФИО собственника	

Собственник (**ФИО собственника**, Адрес проживания собственника, Номер телефона собственника);

Фирма (**Государственный регистрационный № фирмы**, Юридический адрес фирмы, Название фирмы, Тип фирмы, *ФИО собственника*);

Офис (**Адрес офиса**, Площадь офиса, Количество персонала офиса, *Государственный регистрационный № фирмы*);

Предметной областью разрабатываемой базы системы является информация об унитарных предприятиях, зарегистрированных в России. Информация о предприятиях включает в себя информацию о собственнике, о самой фирме, об офисе. Необходимо обеспечить возможность внесения, изменения или удаления данных в базе и проведение различных поисковых операций: поиск по названию фирмы, поиск по ФИО предпринимателя.

1. Моделирование

Модель – это визуальное представление структуры данных. Модель может включать в себя следующие объекты: таблицы и отношения, которые используются обязательно, и дополнительные (например, изображения, записи) – для обеспечения лучшего "понимания" структуры модели.

Для создания модели необходимо переключится в DesignMode, выбрав меню **Display - DesignMode**.

Пользовательский интерфейс делает создание модели базы данных очень легким.

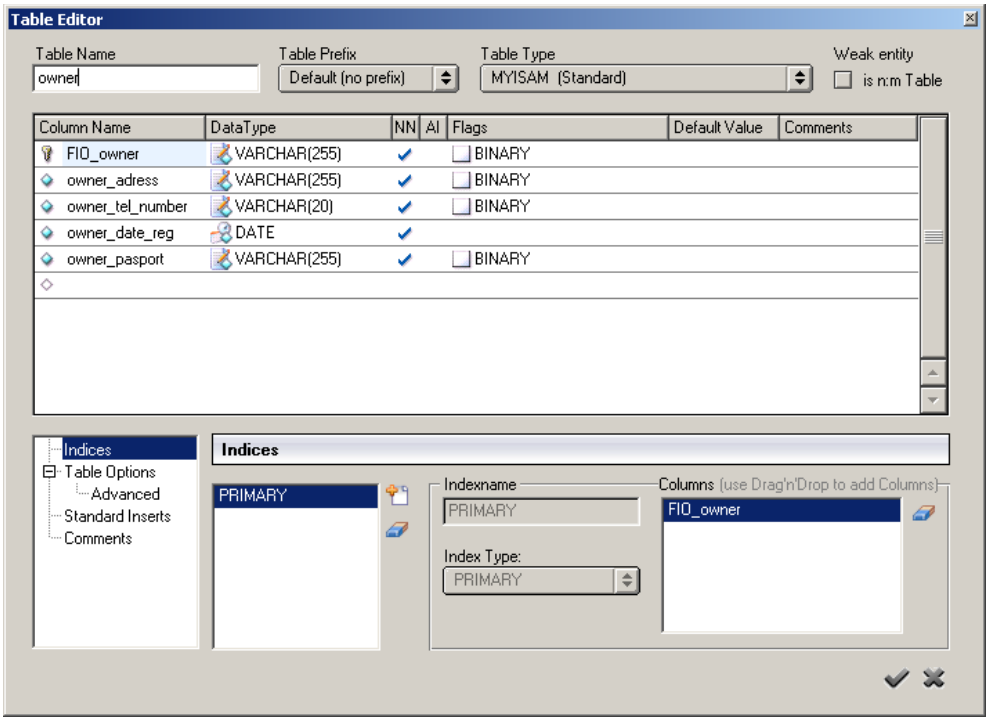
DBDesigner 4 поддерживает Multible Document Interface(MDI), который позволяет открывать неограниченное число моделей одновременно. При работе, вы можете переключаться между моделями, копируя команды и объекты, чтобы обмениваться ими между моделями.

1) Создание таблиц

В левой части холста находится панель инструментов. Необходимо нажать на этой панели кнопку  и указать место на холсте, где будет располагаться новая сущность. Появится прямоугольное изображение пустой сущности. Чтобы задать

атрибуты сущности, необходимо два раз щелкнуть на изображении сущности. В появившемся окне можно задать название сущности, а также атрибуты этой сущности.

В данной лабораторной работе область представляет собой информацию о фирмах, их владельцах, и о офисах, где располагаются эти фирмы. Для отображения личной информации о собственниках была создана сущность **owner**. Структура сущности **owner**показана на рисунке:

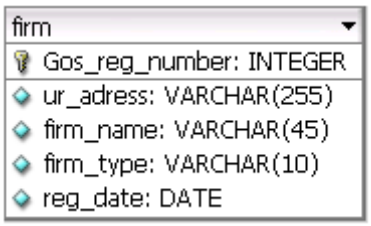


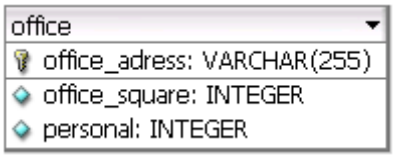
Для этой сущности введены следующие атрибуты:

- ФИО собственника (FIO_owner) (ключевой атрибут),
- Адрес проживания собственника (owner_adress),
- Телефонный номер собственника (owner_tel_number),
- Дата регистрации (owner_date_reg),
- Паспортные данные(owner_pasport)

Флаг в поле NN означает, что содержимое данного поля не может быть нулевым (NotNull). Флаг в поле AI означает, что значение данного поля в каждой следующей строке увеличивается на 1 (AutoIncrement). Иконка напротив имени атрибута означает, что этот атрибут является ключевым.

Были созданы еще две сущности: фирма (firm) и офис (office)





Атрибуты сущности **firm**:

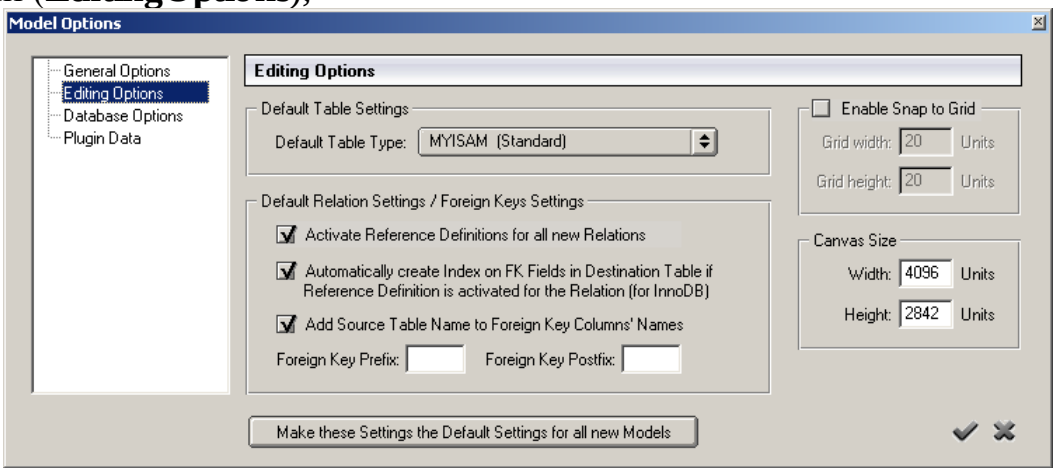
- Государственный регистрационный номер фирмы (Gos_reg_number) (ключевой атрибут),
- Юридический адрес фирмы (ur_adress),
- Название фирмы (firm_name),
- Дата регистрации фирмы (reg_date)

Атрибуты сущности **office**:

- Адрес офиса (office_adress) (ключевой атрибут),
- Площадь офиса (office_square),
- Количество персонала (personal).

1) Формирование отношений

Связь между сущностями определяет связь между будущими таблицами. Для этого необходимо поставить флаг напротив после всех полей в разделе параметры отношений и внешних ключей (раздел **DefaultRelationSettings / ForeignKeysSettings**) во вкладке редактирования модели (**EditingOptions**),



Между сущностями имеются следующие связи:

Собственник –> владеет ->**Фирмой**

Собственник может владеть несколькими фирмами, каждая фирма может иметь только одного собственника. Собственник должен иметь хотя бы одну фирму, фирма обязательно должна принадлежать кому-то. Имеем связь 1:m, класс принадлежности O:O.

Фирма -> имеет ->**Офис**

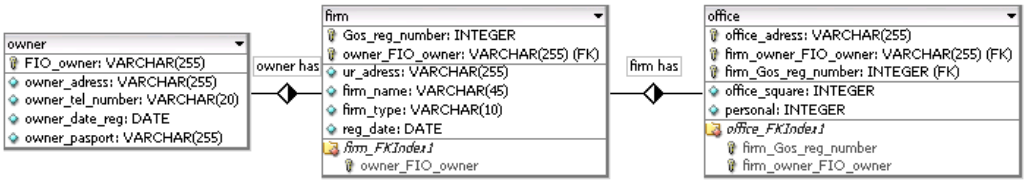
Фирма может иметь несколько офисов, один офис может иметь только одна фирма. Фирма может не иметь ни одного офиса, но офис обязательно должен принадлежать какой-то фирме. Имеем связь 1:m, класс принадлежности H:O.

В программе связи задаются следующим образом.

– Связь 1:1 задается с помощью кнопки  .
– Связь 1:n задается с помощью кнопки  .
– Связь n:m задается с помощью кнопки  .

Задать связь между сущностями можно, нажав на соответствующую кнопку и указав связываемые таблицы. После нажатия кнопки связи, надо нажать на первую таблицу, участвующую в связи, затем на другую. Внешние ключи будут автоматически добавлены в сущности соответственно связи.

Результат связывания сущностей показан на рисунке:



Двойным щелчком по изображению связь можно редактировать свойства связи, такие как название связи и тип связи.

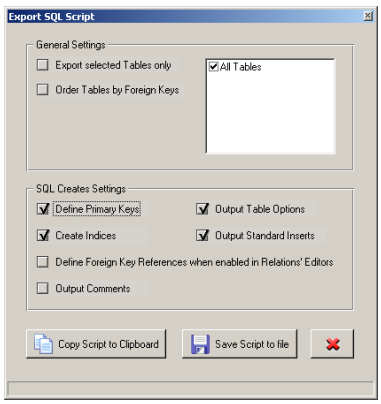
Особенностью программы DBDesigner является то, что в процессе создания ER-диаграмм, отношения будет сформированы автоматически. Это очень удобно при проектировании, т.к. позволяет разработчику не запоминать правила формирования отношений для различных связей между таблицами, а получать все автоматически на основе анализа предметной области.

Связи между таблицами можно корректировать, используя "Редактор связей" (**RelationEditor**), вызываемый двойным щелчком мыши. В "Редакторе связей" можно задать имя связи, изменить ее тип и задать ограничения на данные таблицы при удалении и добавлении в нее данных.

2. Кодирование

DBDesigner позволяет преобразовывать полученную модель в код на языке SQL, который может быть использован для создания базы данных с помощью других средств, например, с помощью MySQL.

Для получения кода необходимо выбрать в меню **File – Export – SQLCreateScript**. Откроется диалоговое окно, представленное на рисунке:



В основных настройках (**GeneralSettings**) можно назначить экспортировать в SQL код только выделенные таблицы или экспортировать все таблицы модели, также можно задать упорядочить таблицы по внешним ключам.

- **Exportselectedtablesonly** – кодировать только выбранные таблицы

- **OrderTablesbyForeignKeys** – позволяет изменить порядок кодирования

В настройках SQL кода (**SQLCreatesSettings**) можно настроить параметры связанные с первичными ключами и внешними ключами, а также задать настройки относительно индексов.

CopyScripttoClipboard. Позволяет скопировать SQL код в буфер обмена;

SaveScripttofile. Позволяет сохранить SQL код в файл. Файл сохраняется в формате *.sql. Открыть его можно и в текстовом редакторе "Блокнот"

Выбрав необходимые параметры, необходимо нажать SaveScripttofile. Файл с SQL кодом будет сохранен на диске.

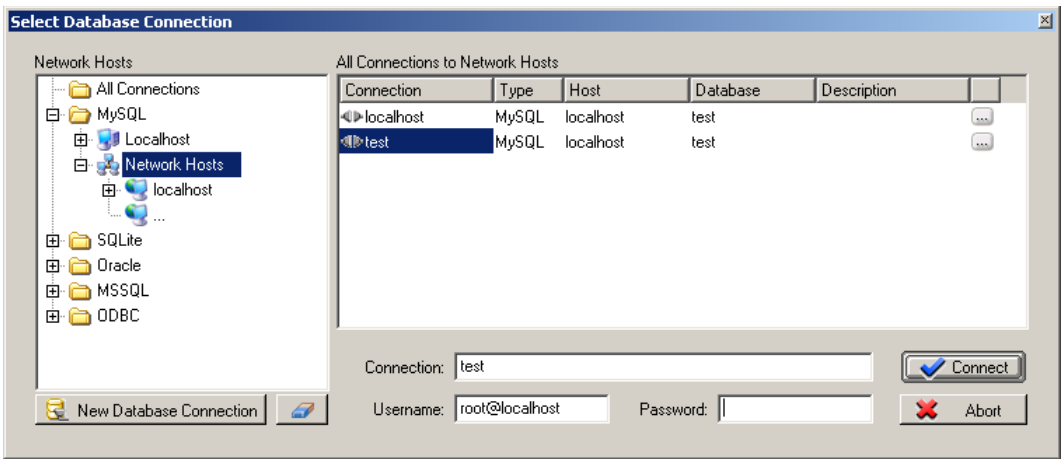
```
CREATETABLEfirm (
Gos_reg_number INTEGER UNSIGNED NOT NULL,
 owner_FIO_owner VARCHAR(255) NOT NULL,
 ur_adress VARCHAR(255) NOT NULL DEFAULT not detected,
 firm_name VARCHAR(45) NOT NULL,
 firm_type VARCHAR(10) NOT NULL,
 reg_date DATE NOT NULL,
 PRIMARY KEY(Gos_reg_number, owner_FIO_owner),
 INDEX firm_FKIndex1(owner_FIO_owner)
);
CREATE TABLE office (
 office_adress VARCHAR(255) NOT NULL,
 firm_owner_FIO_owner VARCHAR(255) NOT NULL,
 firm_Gos_reg_number INTEGER UNSIGNED NOT NULL,
 office_square INTEGER UNSIGNED NOT NULL,
 personal INTEGER UNSIGNED NOT NULL,
 PRIMARY KEY(office_adress, firm_owner_FIO_owner,
firm_Gos_reg_number),
 INDEX office_FKIndex1(firm_Gos_reg_number, firm_owner_FIO_owner)
);
CREATE TABLE owner (
 FIO_owner VARCHAR(255) NOT NULL,
 owner_adress VARCHAR(255) NOT NULL,
 owner_tel_number VARCHAR(20) NOT NULL,
 owner_date_reg DATE NOT NULL,
 owner_pasport VARCHAR(255) NOT NULL,
 PRIMARYKEY(FIO_owner)
);
```

3. Работа с базой данных

DBDesigner позволяет также создавать базу данных на сервере и выполнять с ней различные операции. Это обеспечивается за счет подключения DBDesigner к MySQL серверу, созданию базы данных и установлению синхронизации между базой на сервере и визуальной моделью. Синхронизация – это сравнение визуальной модели и базы данных, находящейся на сервере. В случае внесения изменений в таблицу, изменения связей между таблицами или удаления таблиц в модели, DBDesigner внесет и соответствующие изменения в базу на сервере.

1) Установление соединения с базой данных на сервере

Для занесения базы данных, соответствующей полученной модели, на сервер MySQL, необходимо установить соединение с сервером.



Выполните**Database –> Connect to Database**.

Вокне**Network Hosts**выберите MySQL

В открывшемся списке баз данных, выберите либо существующую базу, либо создать новую, щелкнув два раза по значку "…" и задав имя новой базы.

Введите название соединения (**Connection**), имя пользователя (**Username**) и пароль (**Password**), если они нужны.

В центральном окне находится список серверов баз данных, с которыми велась работа и для которых указаны IP-адрес, тип, размещение и название. Так как в данной работе предполагается, что сервер MySQL находиться на локальном компьютере, то все необходимые параметры будут установлены автоматически. Однако при использовании сети, необходимо знать IP-адрес сервера и иметь доступ на работу.

Нажмите на кнопку **Connect**, после чего соединение с базой будет установлено.

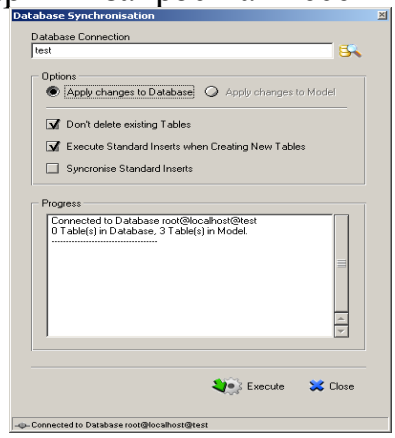
2) Синхронизация

Для синхронизации модели и базы на сервере необходимо:

- Выбрать в меню **Database - DatabaseSynchronisation** и установить соединение с нужной базой.

- В диалоговом окне DatabaseSynchronisation задать необходимые параметры:

  - **ApplychangestoDatabase** – вносить изменения модели в базу
  - **Don'tdeleteexisitingTables** – при использовании этой опции таблицы, удаленные из модели, не будут удалены из базы
  - **Execute Standard Inserts when Creating New Tables** – создаватьстандартныйзапрос на внесение данных в таблицу



Нажать **Execute,** после чего база данных будет занесена на сервер. Также будет выведен отчет и сообщения об ошибках в модели, если они есть.

Проверка получившегося с помощью клиента MySQL:

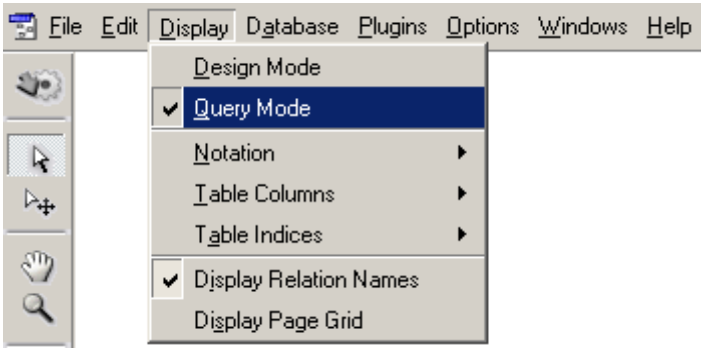
```
mysql> USE test;
Database changed
mysql> SHOW TABLES;
+----------------+
| Tables_in_test |
+----------------+
| firm           |
| office         |
| owner          |
+----------------+
3 rows in set (0.00 sec)

mysql>
```

4. SQL-запросы

DBDesigner также позволяет создавать запросы на языке SQL. Причем код запроса можно либо непосредственно написать, либо использовать готовые шаблоны, в которые необходимо только внести какие-то изменения.

Для работы с запросами необходимо:

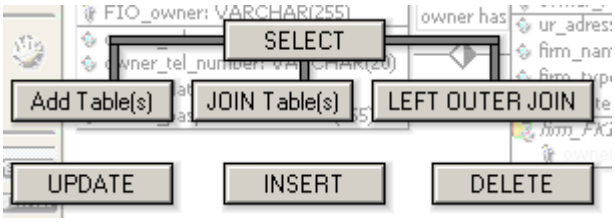
- Переключиться в QueryMode, выбрав в меню **Display ->QueryMode**.



- В меню инструментов слева появятся кнопки, с помощью которых можно выполнить основные запросы.

- Выбрав кнопку (например, SELECT), следует щелкнуть по заголовку таблицы, а затем, не отпуская кнопку мыши, сдвинуть указатель вниз.

- В появившемся меню выбрать нужную операцию.



- Код на языке SQL появиться в нижней части экрана.

```
SELECT *
FROM owner
```

- Нажав на кнопку **ExecuteSQLQuery**, в нижней части экрана можно увидеть результат запроса.

FIO_owner	owner_adress	owner_tel_number	owner_date_reg	owner_pasport
Зубастиков И. А.	Варшавское ш., дом 4	123-123-123	10.09.1990	серия:1111 номер:2121212
Дизайнер Д.Б.	ул. Транзисторная, дом 34	987-987-987	05.11.2002	серия:6543 номер:0987654
Дюнин А.Ц.	ул. Полупроводниковая, дом 76	456-456-456	12.09.2006	серия:2222 номер:4444444
Чебурекин В.И.	ул. Владимирская, дом 56	111-111-111	26.02.1993	серия:5555 номер:2567562

```
SELECT f.owner_FIO_owner, of.office_adress, of.personal
FROM owner o, firm f, office of
WHERE owner_FIO_owner=FIO_owner AND
firm_Gos_reg_number=Gos_reg_number AND
firm_owner_FIO_owner=owner_FIO_owner
```

owner_FIO_owner	office_adress	personal	firm_name	personal_1
Зубастиков И. А.	Протоиерейская ул. Д.	345	ABBYY	345

```
SELECT o.FIO_owner, f.firm_name, o.owner_tel_number,
o.owner_date_reg
FROM owner o, firm f
WHERE owner_FIO_owner=FIO_owner AND
owner_date_reg<'31.12.2002'
```

ORDER BY owner_date_reg

FIO_owner	firm_name	owner_tel_number	owner_date_reg
Зубастиков И. А.	ABBYY	123-123-123	10.09.1990
Дизайнер Д.Б.	DRIVE	987-987-987	05.11.2002
Дюнин А.Ц.	krypnosoft	456-456-456	12.09.2006

```
UPDATE firm
SET reg_date='2006-09-10',
Gos_reg_number=(Gos_reg_number+100000)
WHERE firm_type='OOO'
```

Gos_reg_number	owner_FIO_owner	ur_adress	firm_name	firm_type	reg_date
123456	Зубастиков И. А.	адрес	ABBYY	ЗАО	09.12.1993
345678	Дюнин А.Ц.	ул. Визуальная, д. 76	krypnosoft	ОАО	04.04.1987
754321	Дизайнер Д.Б.	ул. Кодинга, д. 23	DRIVE	ООО	10.09.2006

```
SELECT MIN(o.owner_date_reg) AS 'самаяранняядатарегистрации'
FROM owner o
```

самая ранняя дата регистрации
10.09.1990

DBDesigner предоставляет различные функции для работы с запросами: сохранение кода, внесение изменений в базу и отмена внесенных изменений. Благодаря этим встроенным функциям работа с запросами существенно упрощается.

Кроме того, в программе есть очень удобное средство для внесения данных в таблицу. Щелкнув правой кнопкой мыши по таблице и выбрав в меню EditTableData, можно заносить данные в таблицу или изменять их без использования языка SQL.

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания выполнить корректировку структуры базы данных, применяя принципы нормализации базы данных.

3 Создать схему базы данных с использованием CASE-средств.

4 Преобразовать схему в SQL код

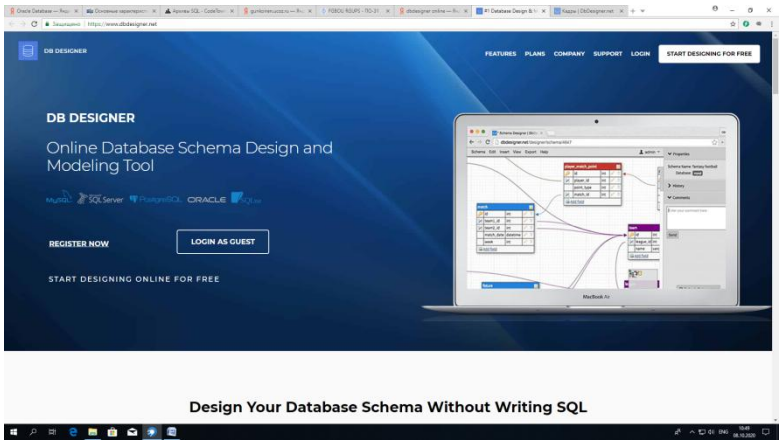
Порядок выполнения работы

Для создания схемы воспользуемся программой DB DESIGNER ONLINE. DBDesigner – это свободно распространяемая CASE-система,

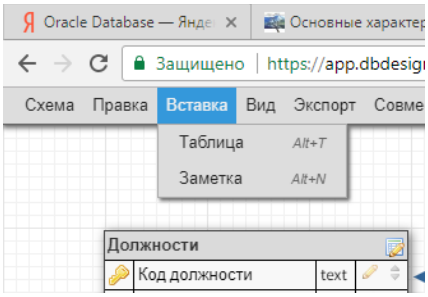
предназначенная для проектирования, моделирования, создания и поддержки информационных систем. Программа может использоваться для Windows 2000/XP, Linux KDE/Gnome и MySQL. DBDesigner позволяет:

– создавать модель проектируемой системы;
– преобразовывать модели системы в SQL-код, который можно использовать для создания базы данных с помощью DBDesigner или другого средства и пр.

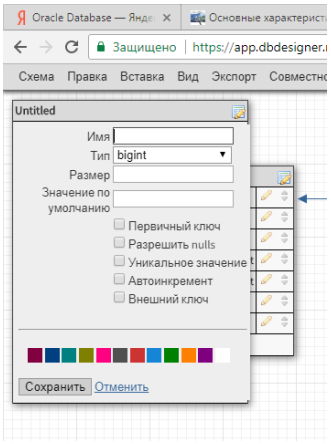
Перед началом работы необходимо зарегистрироваться, указав адрес почтового ящика.



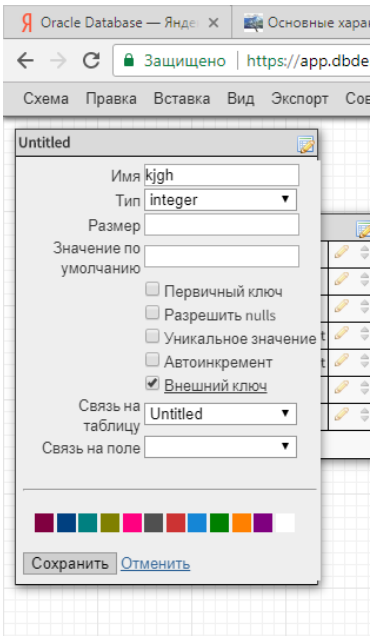
В меню выбираем Вставка/ Таблица.



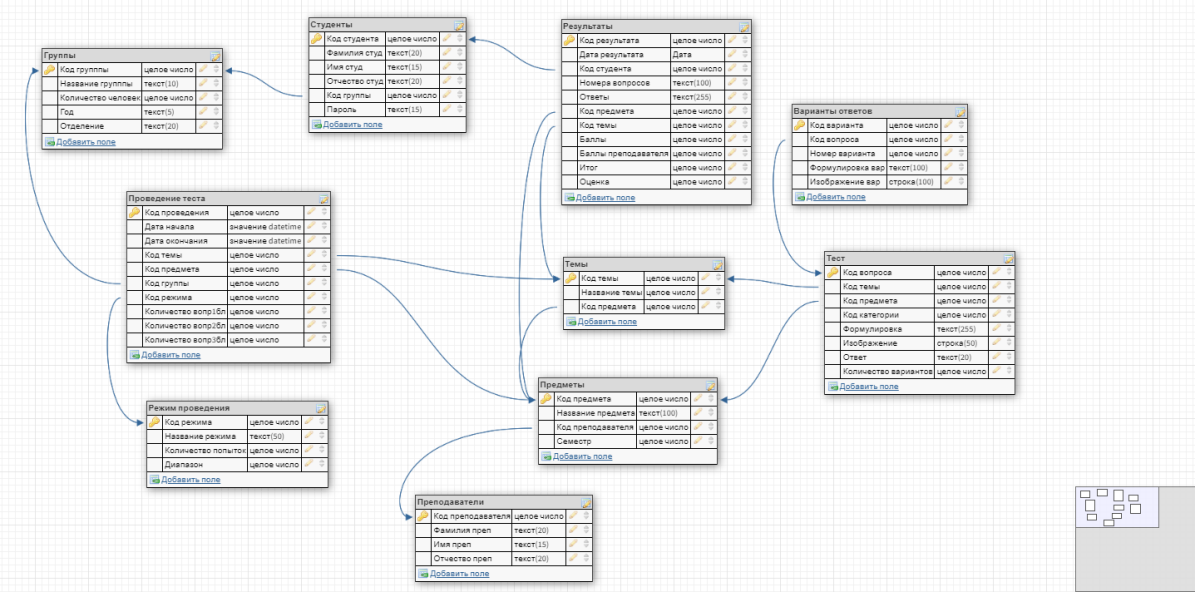
В появившейся области для ввода меняем название таблицы, затем нажимаем Добавить поле, вносим его характеристики.



Если поле является внешним ключом, в открывшемся меню указываем таблицу для связи, поле появится автоматически. После сохранения нового поля, программа сама свяжет таблицы.

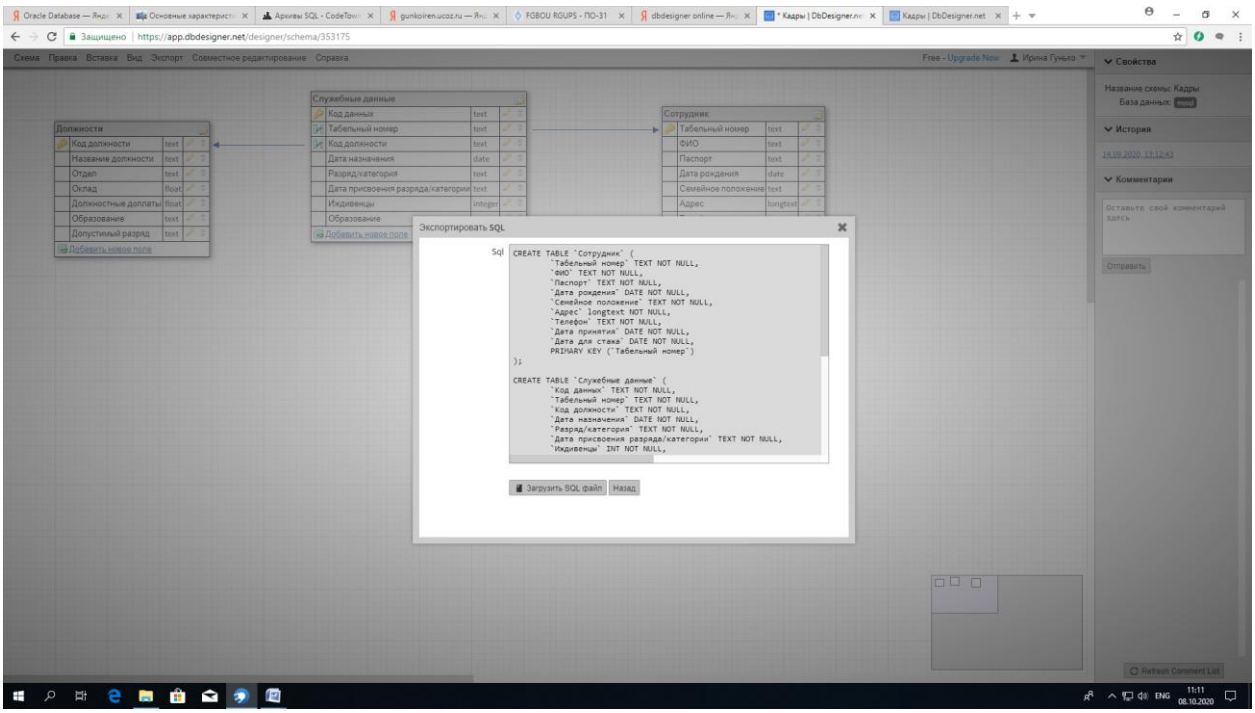


Кроме имени, типа и размера поля желательно задать другие характеристики, например, определить, может ли это поле оставаться пустым или могут ли данные в поле повторяться. Таким образом создать все таблицы базы данных.



Кроме имени, типа и размера поля желательно задать другие характеристики, например, определить, может ли это поле оставаться пустым или могут ли данные в поле повторяться. Таким образом создать все таблицы базы данных.

Для получения SQL кода выбираем в меню раздел Экспорт, и DBDesigner генерирует код создания таблиц.



Этот код можно скопировать и использовать для создания таблиц базы данных.

```
CREATE TABLE `Группы`
(`Код группы` INT NOT NULL,
`Название группы`  CHAR(10) NOT NULL,
`Количество человек` INT,
`Год`  CHAR(5),
`Отделение`  CHAR(20) NOT NULL,
PRIMARY KEY (`Код группы`)
);
CREATE TABLE `Предметы`
(`Код предмета` INT NOT NULL,
`Название предмета`  CHAR(100) NOT NULL,
`Код преподавателя` INT NOT NULL,
`Семестр` INT NOT NULL,
PRIMARY KEY (`Код предмета`)
);

CREATE TABLE `Преподаватели` (
`Код преподавателя` INT NOT NULL,
`Фамилия преп`  CHAR(20) NOT NULL,
`Имя преп`  CHAR(15) NOT NULL,
`Отчество преп`  CHAR(20) NOT NULL,
PRIMARY KEY (`Код преподавателя`)
);
```

```
CREATE TABLE `Проведение теста` (
`Код проведения` INT NOT NULL,
`Дата начала` DATETIME NOT NULL,
`Дата окончания` DATETIME NOT NULL,
`Код темы` INT NOT NULL,
`Код предмета` INT NOT NULL,
`Код группы` INT NOT NULL,
`Код режима` INT NOT NULL,
`Количество вопр1бл` INT NOT NULL,
`Количество вопр2бл` INT NOT NULL,
`Количество вопр3бл` INT NOT NULL,
PRIMARY KEY (`Код проведения`)
);

CREATE TABLE `Режим проведения` (
`Код режима` INT NOT NULL,
`Название режима`  CHAR(50) NOT NULL,
`Количество попыток` INT NOT NULL,
`Диапазон` INT NOT NULL,
PRIMARY KEY (`Код режима`)
);

CREATE TABLE `Темы` (
`Код темы` INT NOT NULL,
`Название темы` INT NOT NULL,
`Код предмета` INT NOT NULL,
PRIMARY KEY (`Код темы`)
);

CREATE TABLE `Результаты` (
`Код результата` INT NOT NULL,
`Дата результата` DATE NOT NULL,
`Код студента` INT NOT NULL,
`Номера вопросов`  CHAR(100) NOT NULL,
`Ответы`  CHAR(255) NOT NULL,
`Код предмета` INT NOT NULL,
`Код темы` INT NOT NULL,
`Баллы` INT NOT NULL,
`Баллы преподавателя` INT NOT NULL,
`Итог` INT NOT NULL,
`Оценка` INT NOT NULL,
PRIMARY KEY (`Код результата`)
);

CREATE TABLE `Тест` (
```

```
`Код вопроса` INT NOT NULL,
`Код темы` INT NOT NULL,
`Код предмета` INT NOT NULL,
`Код категории` INT NOT NULL,
`Формулировка` CHAR(255) NOT NULL,
`Изображение` VARCHAR(50),
`Ответ` CHAR(20),
`Количество вариантов` INT NOT NULL,
PRIMARY KEY (`Код вопроса`)
);

CREATE TABLE `Студенты` (
`Код студента` INT NOT NULL,
`Фамилия студ` CHAR(20) NOT NULL,
`Имя студ` CHAR(15) NOT NULL,
`Отчество студ` CHAR(20) NOT NULL,
`Код группы` INT NOT NULL,
`Пароль` CHAR(15) NOT NULL,
PRIMARY KEY (`Код студента`)
);

CREATE TABLE `Варианты ответов` (
`Код варианта` INT NOT NULL,
`Код вопроса` INT NOT NULL,
`Номер варианта` INT NOT NULL,
`Формулировка вар` CHAR(100) NOT NULL,
`Изображение вар` VARCHAR(100) NOT NULL,
PRIMARY KEY (`Код варианта`)
);

ALTER TABLE `Предметы` ADD CONSTRAINT `Предметы_fk0` FOREIGN KEY (`Код преподавателя`) REFERENCES `Преподаватели`(`Код преподавателя`);

ALTER TABLE `Проведение теста` ADD CONSTRAINT `Проведение теста_fk0` FOREIGN KEY (`Код темы`) REFERENCES `Темы`(`Код темы`);

ALTER TABLE `Проведение теста` ADD CONSTRAINT `Проведение теста_fk1` FOREIGN KEY (`Код предмета`) REFERENCES `Предметы`(`Код предмета`);

ALTER TABLE `Проведение теста` ADD CONSTRAINT `Проведение теста_fk2` FOREIGN KEY (`Код группы`) REFERENCES `Группы`(`Код группы`);
```

ALTER TABLE `Проведение теста` ADD CONSTRAINT `Проведение теста_fk3` FOREIGN KEY (`Код режима`) REFERENCES `Режим проведения`(`Код режима`);

ALTER TABLE `Темы` ADD CONSTRAINT `Темы_fk0` FOREIGN KEY (`Код предмета`) REFERENCES `Предметы`(`Код предмета`);

ALTER TABLE `Результаты` ADD CONSTRAINT `Результаты_fk0` FOREIGN KEY (`Код студента`) REFERENCES `Студенты`(`Код студента`);

ALTER TABLE `Результаты` ADD CONSTRAINT `Результаты_fk1` FOREIGN KEY (`Код предмета`) REFERENCES `Предметы`(`Код предмета`);

ALTER TABLE `Результаты` ADD CONSTRAINT `Результаты_fk2` FOREIGN KEY (`Код темы`) REFERENCES `Темы`(`Код темы`);

ALTER TABLE `Тест` ADD CONSTRAINT `Тест_fk0` FOREIGN KEY (`Код темы`) REFERENCES `Темы`(`Код темы`);

ALTER TABLE `Тест` ADD CONSTRAINT `Тест_fk1` FOREIGN KEY (`Код предмета`) REFERENCES `Предметы`(`Код предмета`);

ALTER TABLE `Студенты` ADD CONSTRAINT `Студенты_fk0` FOREIGN KEY (`Код группы`) REFERENCES `Группы`(`Код группы`);

ALTER TABLE `Варианты ответов` ADD CONSTRAINT `Варианты ответов_fk0` FOREIGN KEY (`Код вопроса`) REFERENCES `Тест`(`Код вопроса`);

Содержание отчета

1 Название работы
2 Цель работы
3 Перечень технических средств обучения
4 Порядок выполнения работы
5 Вывод

Варианты заданий

Варианты заданий представлены в практической работе № 39

Используемая литература

– Г.Н.Федорова Основы проектирования баз данных. М.: Академия, 2020
– Г.Н.Федорова Разработка, администрирование и защита баз данных. М.: Академия, 2018