Содержание

Введение

Предлагаемое пособие содержит задания, которые могут быть использованы для организации лабораторных работ, а также при самостоятельном изучении языка структурированных запросов SQL. В лабораторных работах предполагается использование СУБД MySQL, как программного продукта, не требующего лицензирования. В пособии рассматриваются следующие разделы:

1. Основы работы в MySQL;
2. Команда отбора данных;
3. Использование условий;
4. Агрегированные функции;
5. Использование вычисляемых полей;
6. Сортировка записей в наборе данных;
7. Строковые и числовые функции;
8. Условные выражения с оператором CASE;
9. Простые и связанные подзапросы;
10. Внутренние и внешние соединения;
11. Добавление, удаление и редактирование записей;
12. Связь сторонних приложений с базой данных MySQL (на примере Delphi).

В лабораторных работах предлагается выполнить запросы для уже существующей базы данных, которая заранее размещается на сервере. Такое решение позволяет использовать базу, имеющую сложную структуру и большое количество записей. Для формирования навыков самостоятельного создания баз данных в MySQL, в первой лабораторной на простых примерах описаны принципы работы организации диалога с этой СУБД. Каждая лабораторная содержит краткий теоретический материал, примеры и задания для самостоятельной работы.

Лабораторная работа №1

Разработка базы данных. Основы работы в MySQL

Цель работы: научится производить нормализацию структуры базы данных; научиться создавать базы данных в оболочке MySQL.
Ключевые слова: нормализация, типы данных, создание базы данных, создание таблиц, добавление записей, импортирование и экспортирование базы данных.
Теоретический материал:

1. Нормализация данных

Главная цель нормализации базы данных - устранение избыточности и дублирования информации.
Нормализация – это формальный метод анализа отношений на основе их первичного ключа и существующих функциональных зависимостей.
Отношение – таблица, состоящая из столбцов и строк.
Атрибут – именованный столбец отношения.
Функциональная зависимость – связь между атрибутами отношения [6].

Пример нормализации базы данных

Пусть данные о продажах представлены накладными.

Продажи
Накладная №237 от 06.09.2011 г.

Клиент	Товар	Количество	Цена	Сумма
Иванов	Хлеб	2	24,50 р.	49,00 р.
Петров	Молоко	3	30,00 р.	90,00 р.
ОАО «Рога и копыта»	Хвосты	25	2,00 р.	50,00 р.
ЗАО «111»	Молоко	1	30,00 р.	30,00 р.
Сидоров	Хлеб	3	24,50 р.	73,50 р.

I нормальная форма
Процесс выравнивания.
Для каждого значения должно быть единственное соответствующее значение из неповторяющихся групп [6].

Продажи

Накладная №	Дата	Клиент	Товар	Количество	Цена	Сумма
237	06.09.2011 г.	Иванов	Хлеб	2	24,50 р.	49,00 р.
237	06.09.2011 г.	Петров	Молоко	3	30,00 р.	90,00 р.
237	06.09.2011 г.	ОАО «Рога и копыта»	Хвосты	25	2,00 р.	50,00 р.
237	06.09.2011 г.	ЗАО «111»	Молоко	1	30,00 р.	30,00 р.
237	06.09.2011 г.	Сидоров	Хлеб	3	24,50 р.	73,50 р.

Повторяющиеся группы изымаются и помещаются в отдельные отношения с копией первичного ключа исходного отношения.

Дата

Код продажи	Накладная №	Дата
1	237	06.09.2011 г.

Продажи

Клиент	Товар	Количество	Цена	Сумма	Код продажи
Иванов	Хлеб	2	24,50 р.	49,00 р.	1
Петров	Молоко	3	30,00 р.	90,00 р.	1
ОАО «Рога и копыта»	Хвосты	25	2,00 р.	50,00 р.	1
ЗАО «111»	Молоко	1	30,00 р.	30,00 р.	1
Сидоров	Хлеб	3	24,50 р.	73,50 р.	1

II нормальная форма

Вторая нормальная форма требует, чтобы отношение находилось в первой нормальной форме и неключевые атрибуты отношений зависели от первичного ключа в целом (полная зависимость), но не от его части (частичная зависимость) [6].

Клиенты

Код клиента	Клиент
1	Иванов
2	Петров
3	ОАО «Рога и копыта»
4	ЗАО «111»
5	Сидоров

Товары

Код товара	Товар	Цена
1	Хлеб	24,50 р.
2	Молоко	30,00 р.
3	Хвосты	2,00 р.

Продажи

Код клиента	Код товара	Количество	Сумма	Код продажи
1	1	2	24,50 р.	1
2	2	3	30,00 р.	1
3	3	25	2,00 р.	1
4	2	1	30,00 р.	1
5	1	3	24,50 р.	1

III нормальная форма

Чтобы отношение находилось в третьей нормальной форме, необходимо, чтобы неключевые атрибуты в нем не зависели от других неключевых атрибутов, а зависели только от первичного ключа [6].

Клиенты

Код клиента	Клиент
1	Иванов
2	Петров
3	ОАО «Рога и копыта»
4	ЗАО «111»
5	Сидоров

Товары

Код товара	Товар	Цена
1	Хлеб	24,50 р.
2	Молоко	30,00 р.
3	Хвосты	2,00 р.

Продажи

Код клиента	Код товара	Количество	Код продажи
1	1	2	1
2	2	3	1
3	3	25	1
4	2	1	1
5	1	3	1

2. Типы данных

MySQL поддерживает несколько типов столбцов, которые можно разделить на три категории: числовые типы данных, типы данных для хранения даты и времени и символьные (строковые) типы данных. Мы кратко рассмотрим основные типы данных. Более подробно ознакомиться с типами данных можно в дополнительном материале.

В описаниях используются следующие обозначения:

- M - указывает максимальный размер вывода. Максимально допустимый размер вывода составляет 255 символов.
- D - употребляется для типов данных с плавающей точкой и указывает количество разрядов, следующих за десятичной точкой. Максимально возможная величина составляет 30 разрядов, но не может быть больше, чем M-2.

Квадратные скобки ('[' и ']') указывают для типа данных группы необязательных признаков.

Заметьте, что если для столбца указать параметр ZEROFILL, то MySQL будет автоматически добавлять в этот столбец атрибут UNSIGNED.

- **INT**[(M)] [UNSIGNED] [ZEROFILL]

Целое число нормального размера. Диапазон со знаком от -2147483648 до 2147483647. Диапазон без знака от 0 до4294967295.

- **FLOAT**[(M,D)] [UNSIGNED] [ZEROFILL]

Малое число с плавающей точкой обычной точности. Допустимые значения: от -3,402823466E+38 до -1,175494351E-38, 0, и от 1,175494351E-

38 до 3,402823466E+38. Если указан атрибут UNSIGNED, отрицательные значения недопустимы. Атрибут M указывает количество выводимых пользователю знаков, а атрибут D - количество разрядов, следующих за десятичной точкой. Обозначение FLOAT без указания аргументов или запись вида FLOAT(X), где X <=24справедливы для числа с плавающей точкой обычной точности.

- **DATE**

Дата. Поддерживается интервал от '1000-01-01' до '9999-12-31'. MySQL выводит значения DATE в формате 'YYYY-MM-DD', но можно установить значения в столбец DATE, используя как строки, так и числа.

- [NATIONAL] **CHAR**(M) [BINARY]

Строка фиксированной длины, при хранении всегда дополняется пробелами в конце строки до заданного размера. Диапазон аргумента M составляет от 0 до 255 символов (от 1 до 255 в версиях, предшествующих MySQL 3.23). Концевые пробелы удаляются при выводе значения. Если не задан атрибут чувствительности к регистру BINARY, то величины CHAR сортируются и сравниваются как независимые от регистра в соответствии с установленным по умолчанию алфавитом.

Атрибут NATIONAL CHAR (или его эквивалентная краткая форма NCHAR) представляет собой принятый в ANSI SQL способ указания, что в столбце CHAR должен использоваться установленный по умолчанию набор символов (CHARACTER).

- [NATIONAL] **VARCHAR**(M) [BINARY]

Строка переменной длины. **Примечание:** концевые пробелы удаляются при сохранении значения (в этом заключается отличие от спецификации ANSI SQL). Диапазон аргумента M составляет от 0 до 255 символов (от 1 до 255 в версиях, предшествующих MySQL Version 4.0.2). Если не задан атрибут чувствительности к регистру BINARY, то величины VARCHAR сортируются и сравниваются как независимые от регистра.

3. Основные операции с оболочкой MySQL

Доступ к СУБД MySQL

Для доступа к СУБД MySQL во внутренней сети АГПУ необходимо:

- в адресной строке браузера указать адрес: http://mysql.agpu.net/(Рис. 1);

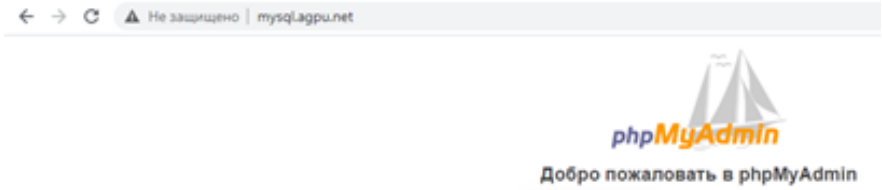

Рис. 1 Адресная строка

- в окне авторизации указать: Пользователь – root (пароль не вводится) (Рис. 2).

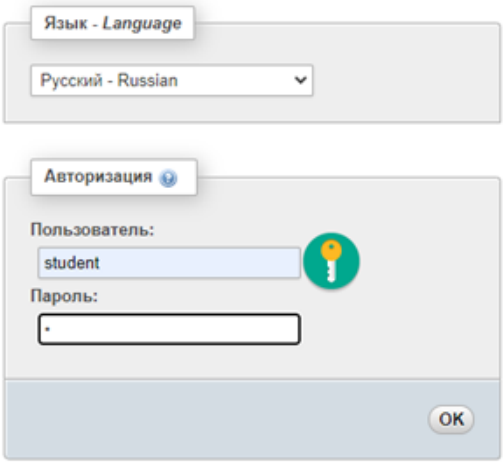

Рис. 2 Авторизация

Создание новой базы данных

Для создания новой базы данных необходимо:
- в строке Новая база данных указать имя будущей базы (только ЛАТИНСКИМИ буквами);
- в строке «Сравнение» указать utf8_general_ci (необходимо для корректного отображения кириллицы) (Рис. 3).

Рис. 3 Создание новой базы данных

- нажать кнопку «Создать».

Создание новой таблицы в базе данных

Для создания новой таблицы в базе данных необходимо:
- указать имя будущей таблицы в строке «Создать новую таблицу в БД»;
- указать количество полей (столбцов) таблицы;
- нажать кнопку «Пошел» (Рис. 4).

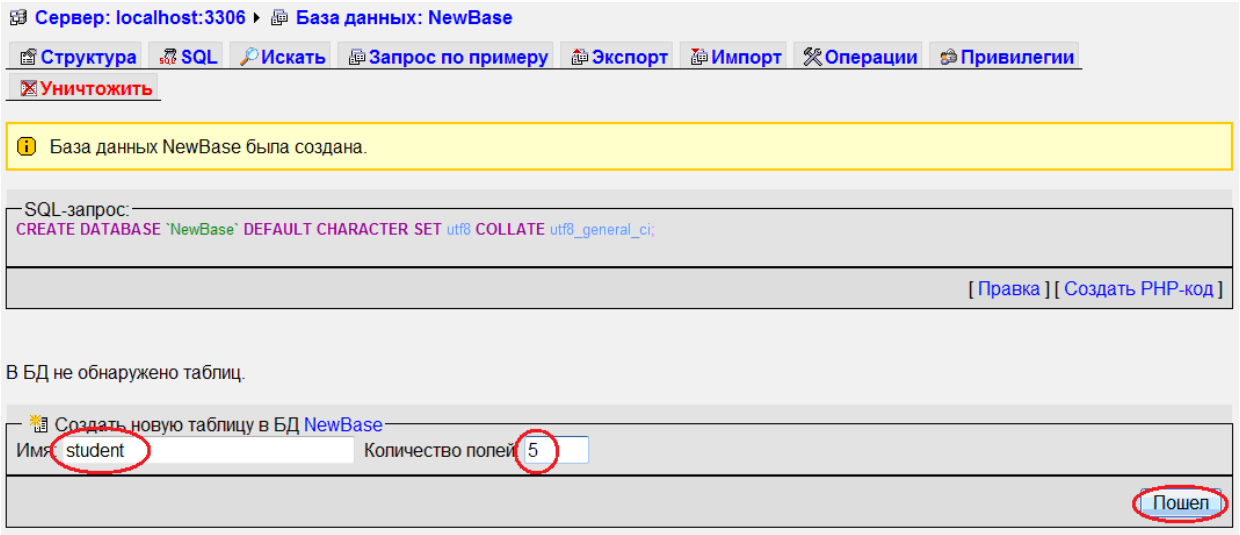

Рис. 4 Создани новой таблицы

- далее необходимо указать названия полей таблицы, типы полей и длины/значения;
- нажать кнопку «Сохранить» (Рис. 5).

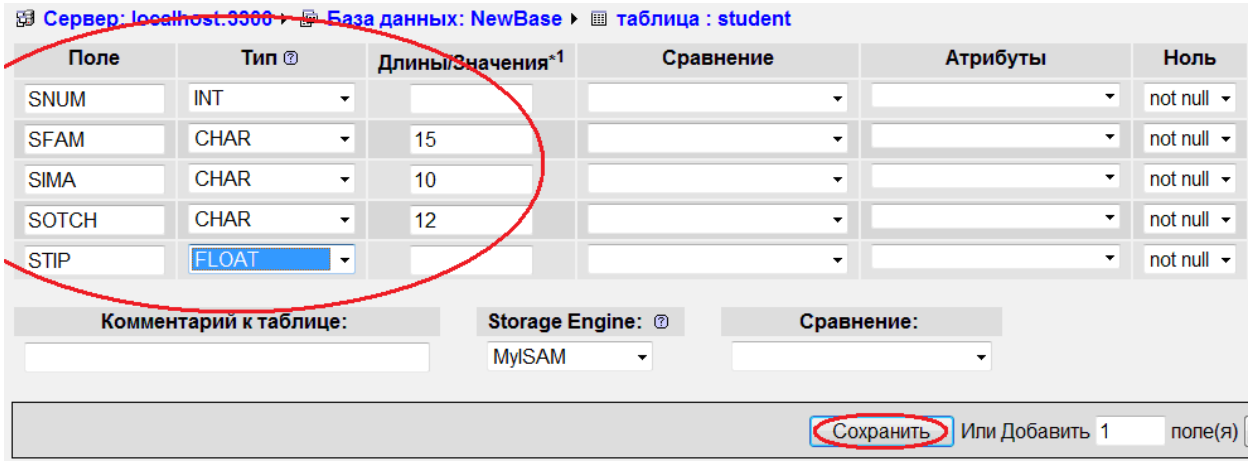

Рис.5 Определение типов данный

- признаком успешного создания таблицы является сообщение: «Таблица была создана» Рис. 6;

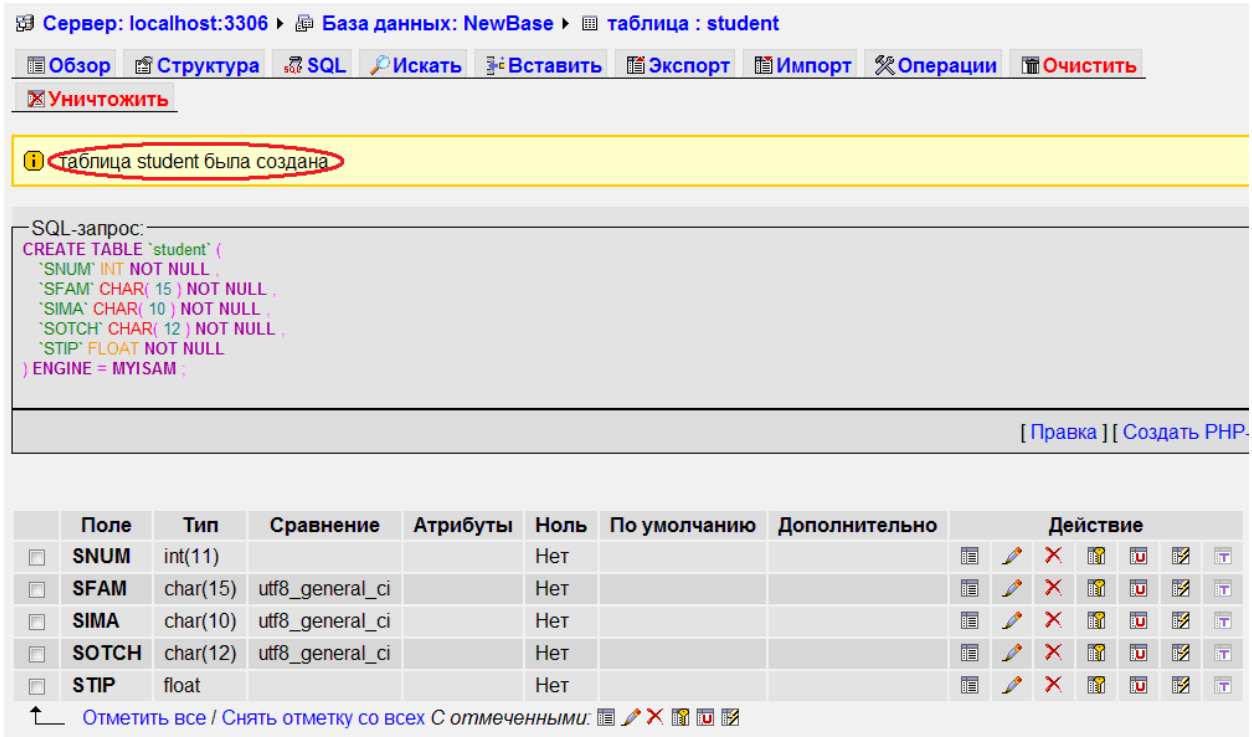

Рис. 6 Признак успешного создания таблицы

Заполнение таблицы данными

Для заполнения таблицы данными необходимо:
- выбрать вкладку «Вставить»;
- ввести нужные значения;
- нажать кнопку «Пошел» (Рис. 7);

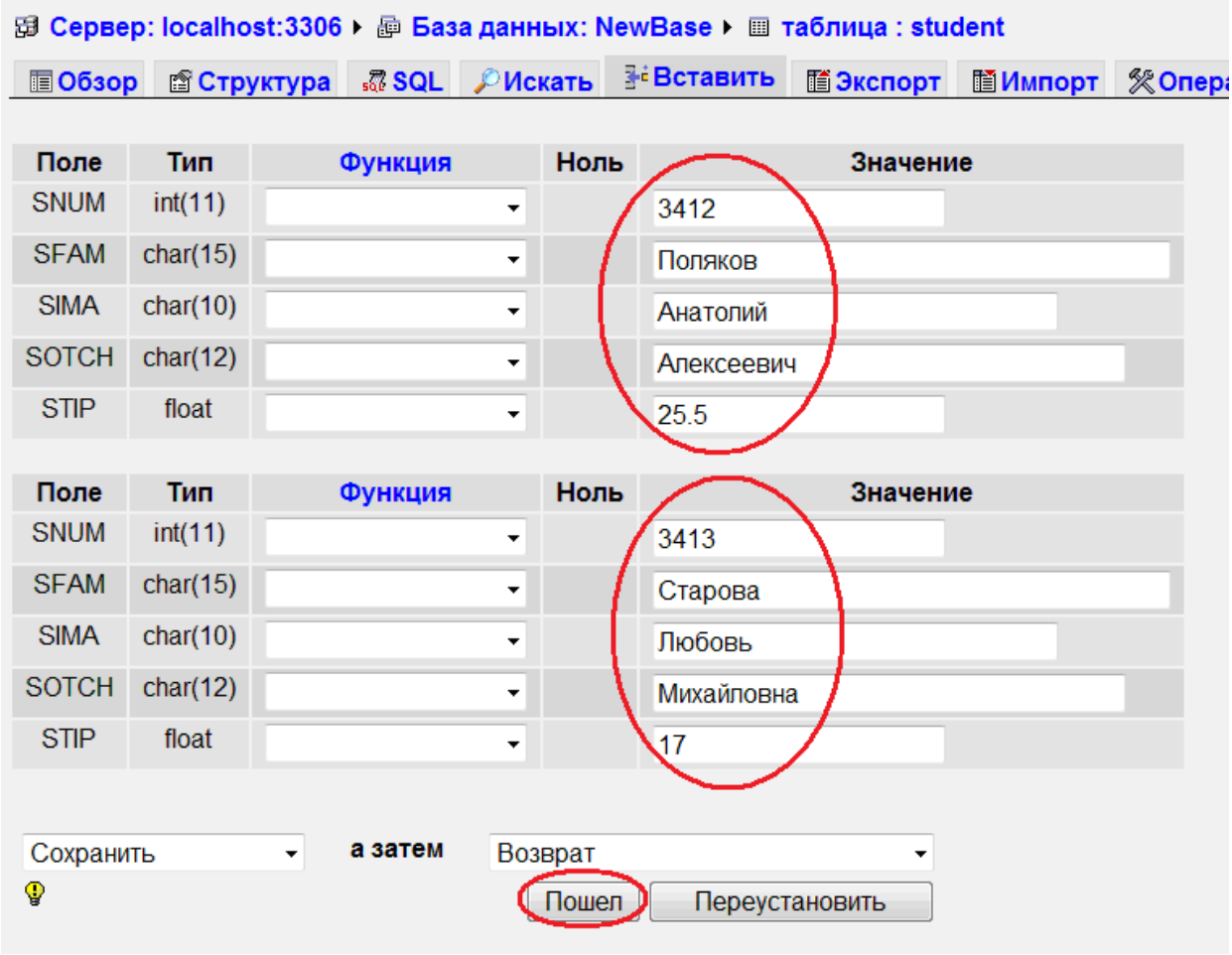

Рис. 7 Заполнение таблицы данными

- при вводе данных типа FLOAT дробная часть отделяется точкой (Рис. 8);

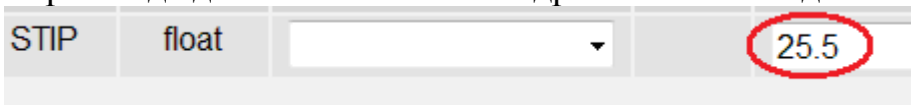

Рис. 8 Ввод дробных чисел

- при вводе данных типа DATE можно воспользоваться либо календарем, либо ввести дату вручную в формате год-месяц-число, например, 1984-04-01 (Рис. 9);

Рис. 9 Ввод даты

- для просмотра введенных данных необходимо выбрать вкладку «Обзор» (Рис. 10);

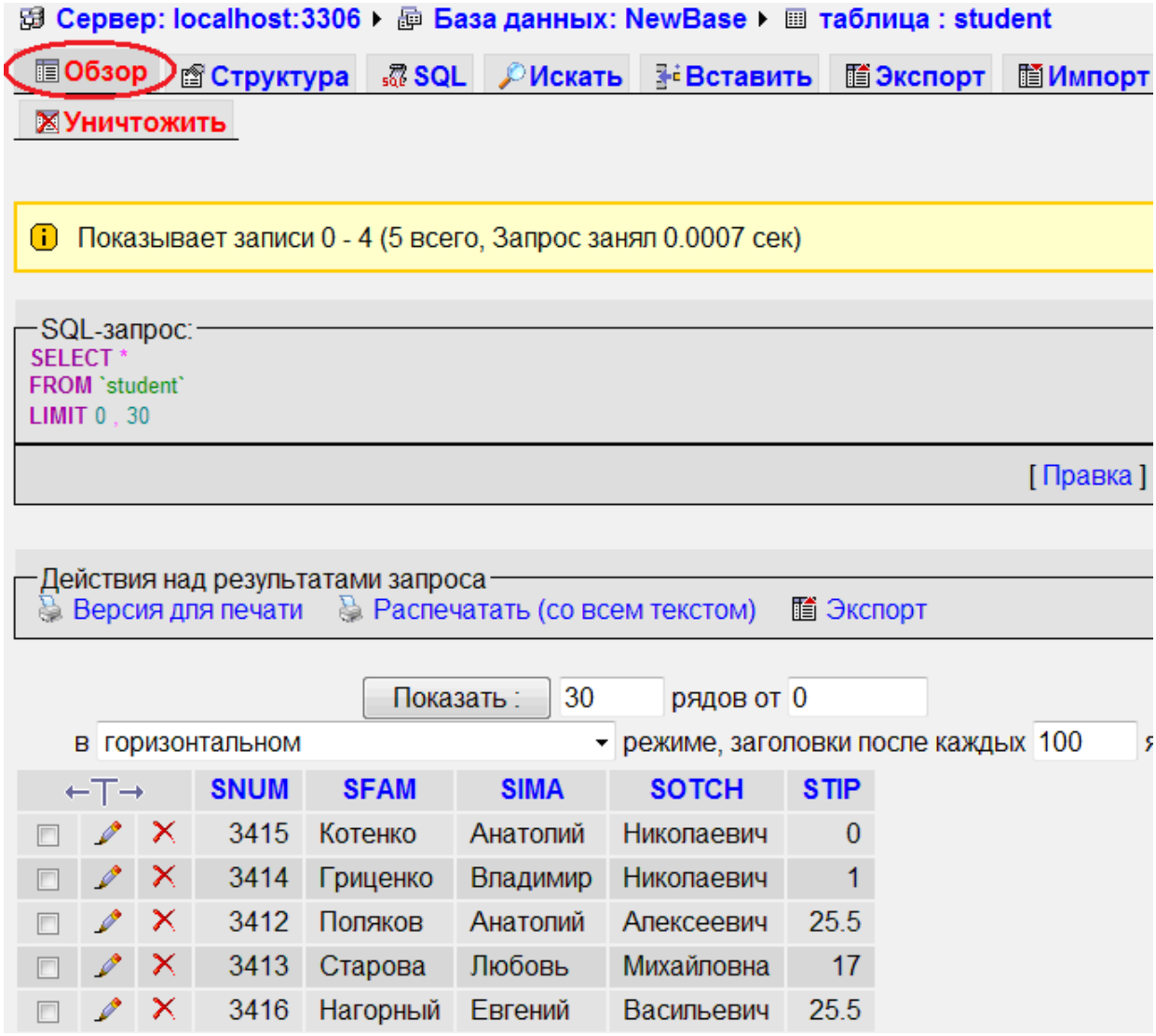

Рис. 10 Внешний вид таблицы

- для исправления записи используется кнопка «Редактировать»  ;

- для удаления записи используется кнопка «Удалить»  ;

Выполнение запросов

Для выполнения запроса к базе данных необходимо:
- выбрать вкладку «SQL»;
- в поле «Выполнить SQL запросы на БД» ввести нужный запрос, например, SELECT SFAM FROM student WHERE STIP=0;
- Нажать кнопку «Пошел» (Рис. 11)
.

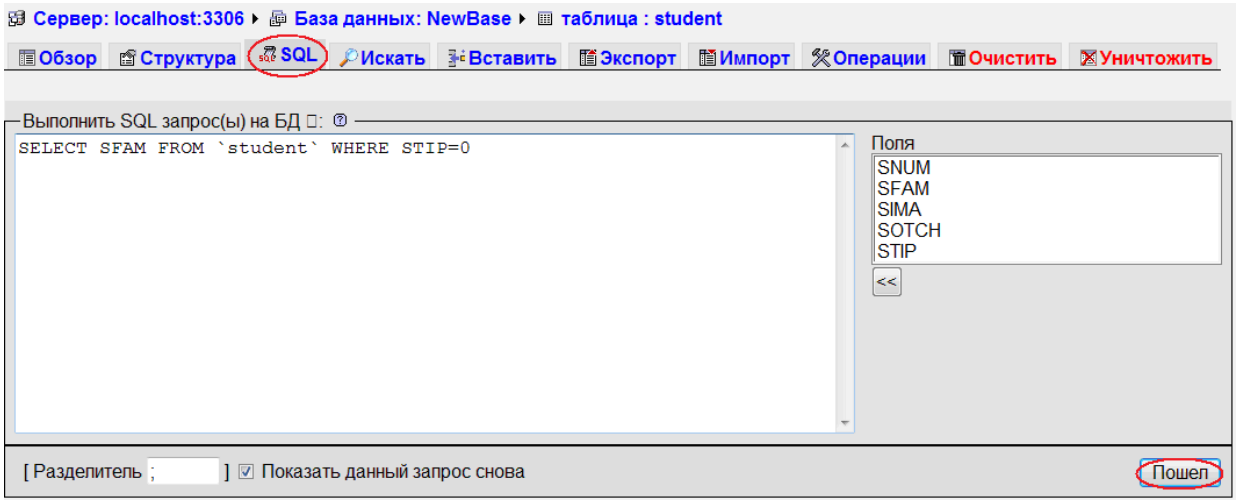

Рис. 11 Создание запросов

- в результате отобразится набор данных, соответствующий запросу (Рис. 12).

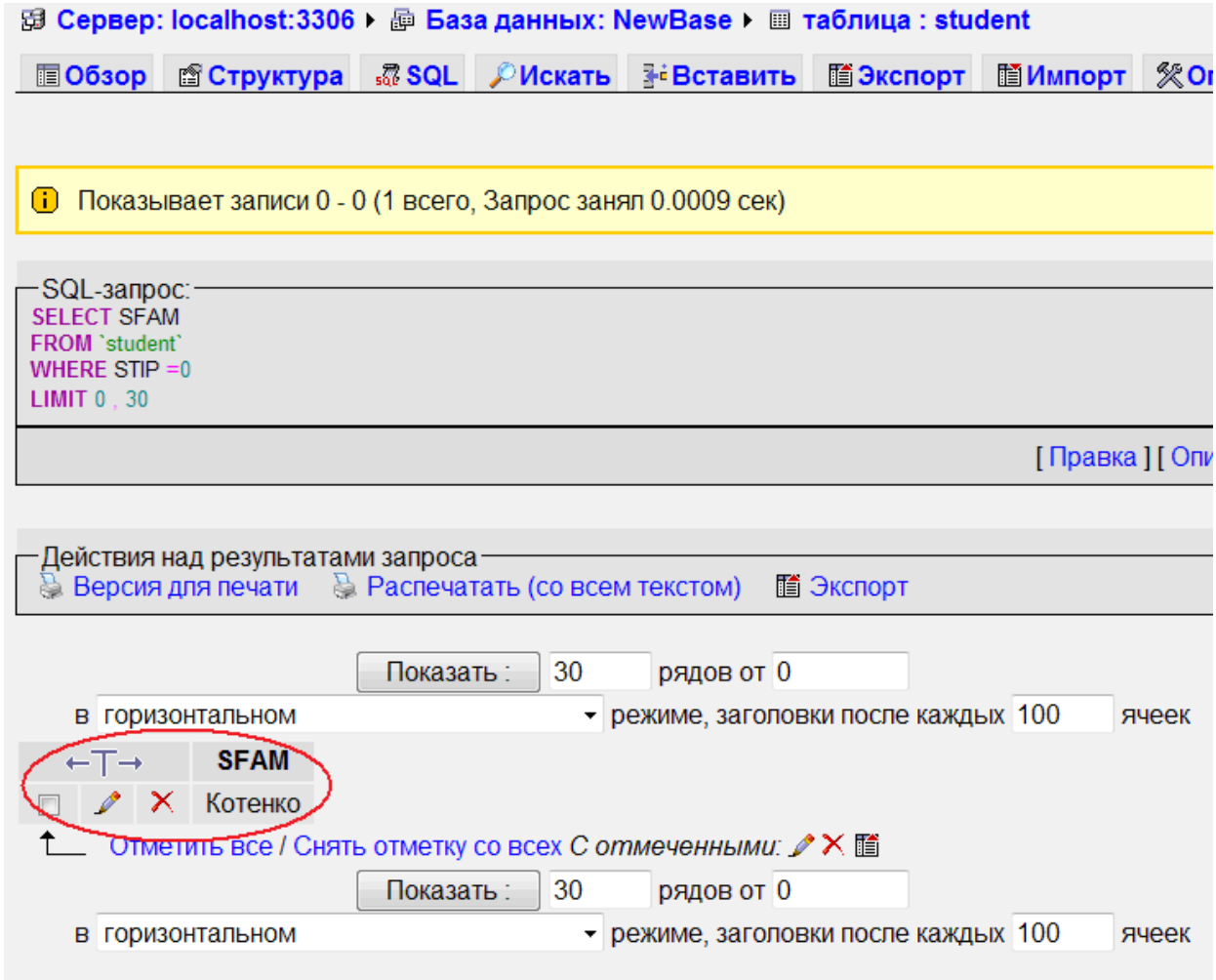

Рис. 12 Результат запроса

В MySQL имеется возможность экспортировать/импортировать БД для переноса на другую СУБД.

Экспорт БД

- выбрать вкладку «Экспорт» (Рис. 13)
;

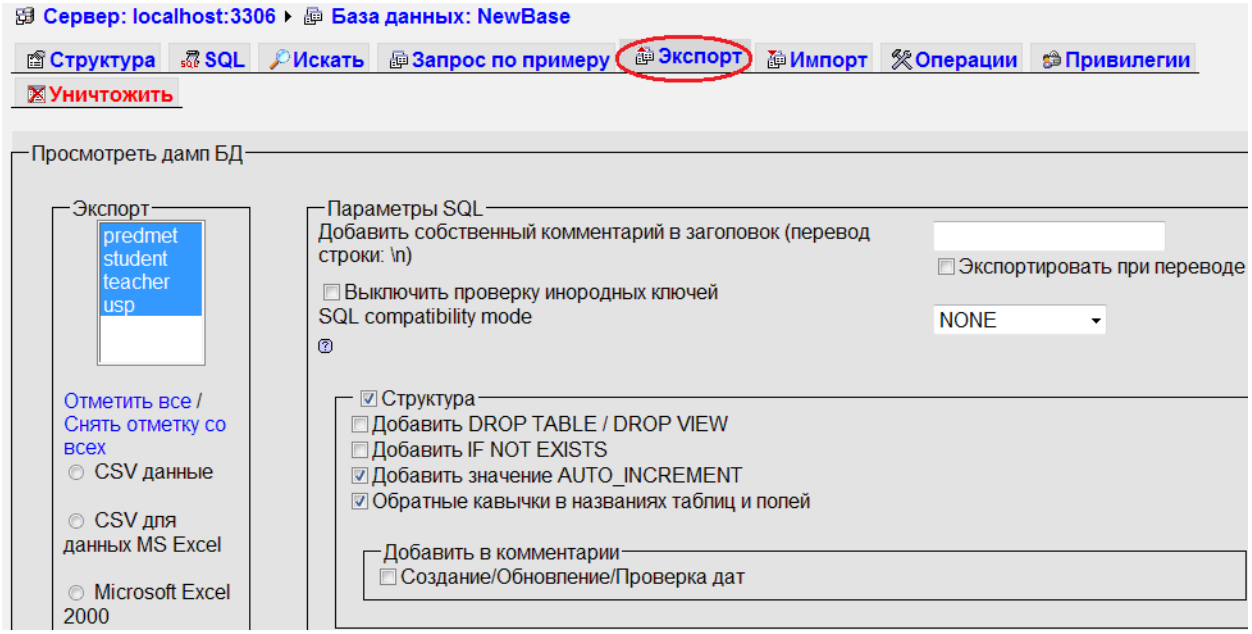

Рис. 13 Экспортирование данных

- указать в левой области тип создаваемого файла – «SQL» (Рис. 14);

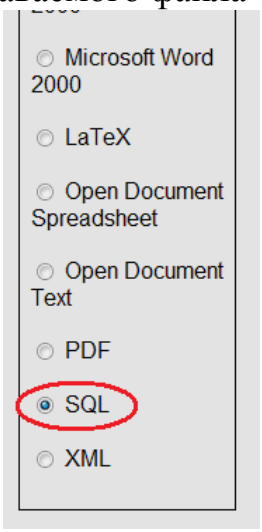

Рис.14 Выбор типа файла

- установить галочку «Сохранить как файл» (Рис. 15):

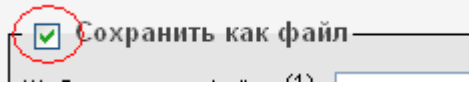

Рис. 15 Указание необходимости сохранить файл

- нажать кнопку «Ok».

Импорт БД

- создать новую БД;
- выбрать вкладку «Импорт»;
- с помощью кнопки «Обзор» загрузить созданный ранее *.sql файл;
- нажать кнопку «Пошел» (Рис. 16);

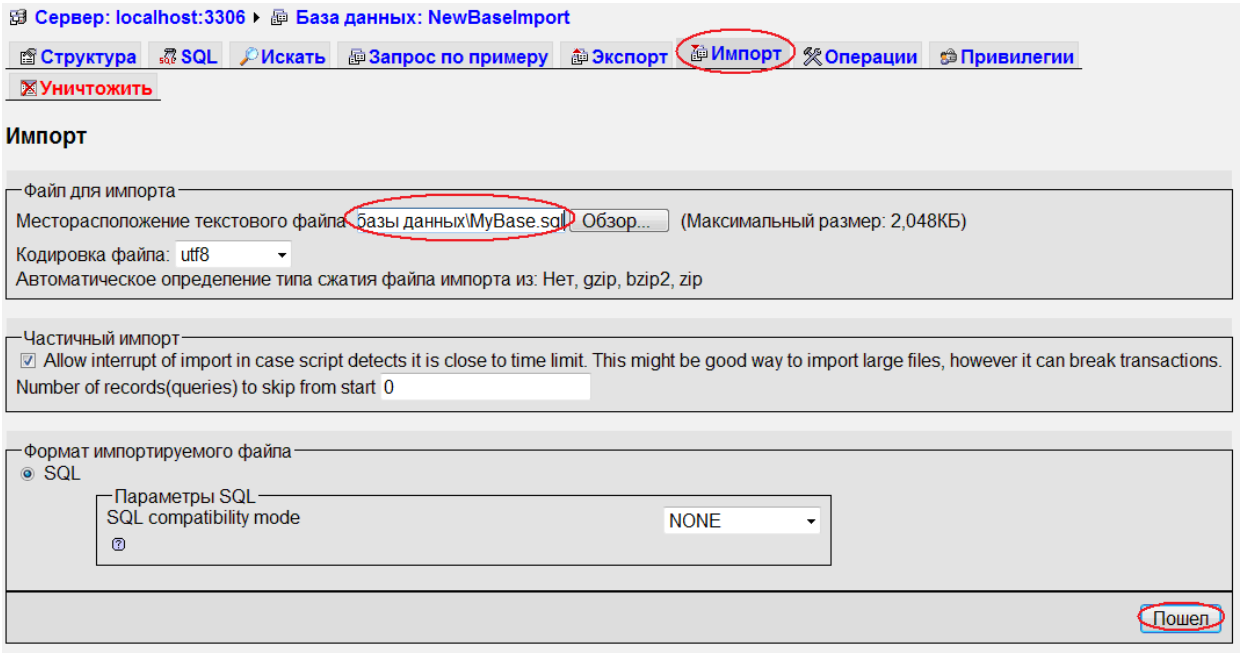

Рис. 16 Импотрирование данных

- признаком успешного завершения операции является сообщение: «Импорт успешно завершен» (Рис. 17).

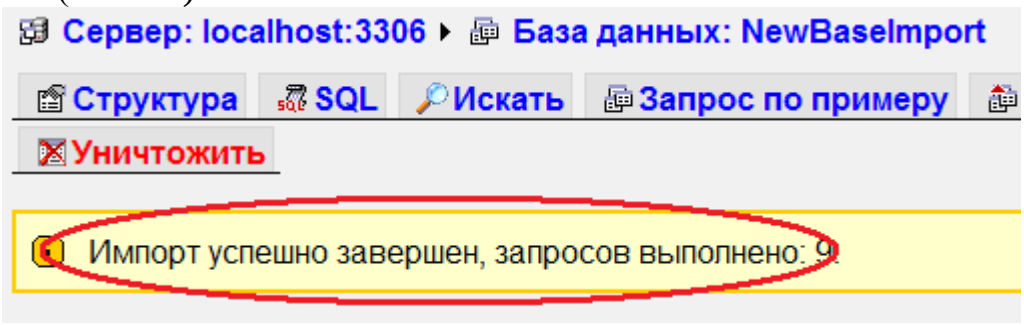

Рис. 17 Признак успешного завершения операции

Задание:

Привести базу данных к III нормальной форме. Создать базу данных в MySql и заполнить её предложенной информацией

Код сотрудника	ФИО	Должность	Номер отдела	Наименование отдела	Квалификация
7513	Иванов Иван Иванович	Программист	128	Отдел проектирования	C, Java
9842	Сергеева Светлана Сергеевна	Администратор БД	42	Финансовый отдел	DB2
6651	Петров Петр Петрович	Программист	128	Отдел проектирования	VB, Java
9006	Николаев Николай Николаевич	Системный администратор	128	Отдел проектирования	Windows, Linux

В дальнейшем лабораторные работы проводятся на примере базы данных STUDENT. Схема базы данных приведена на рисунке (Рис. 18).

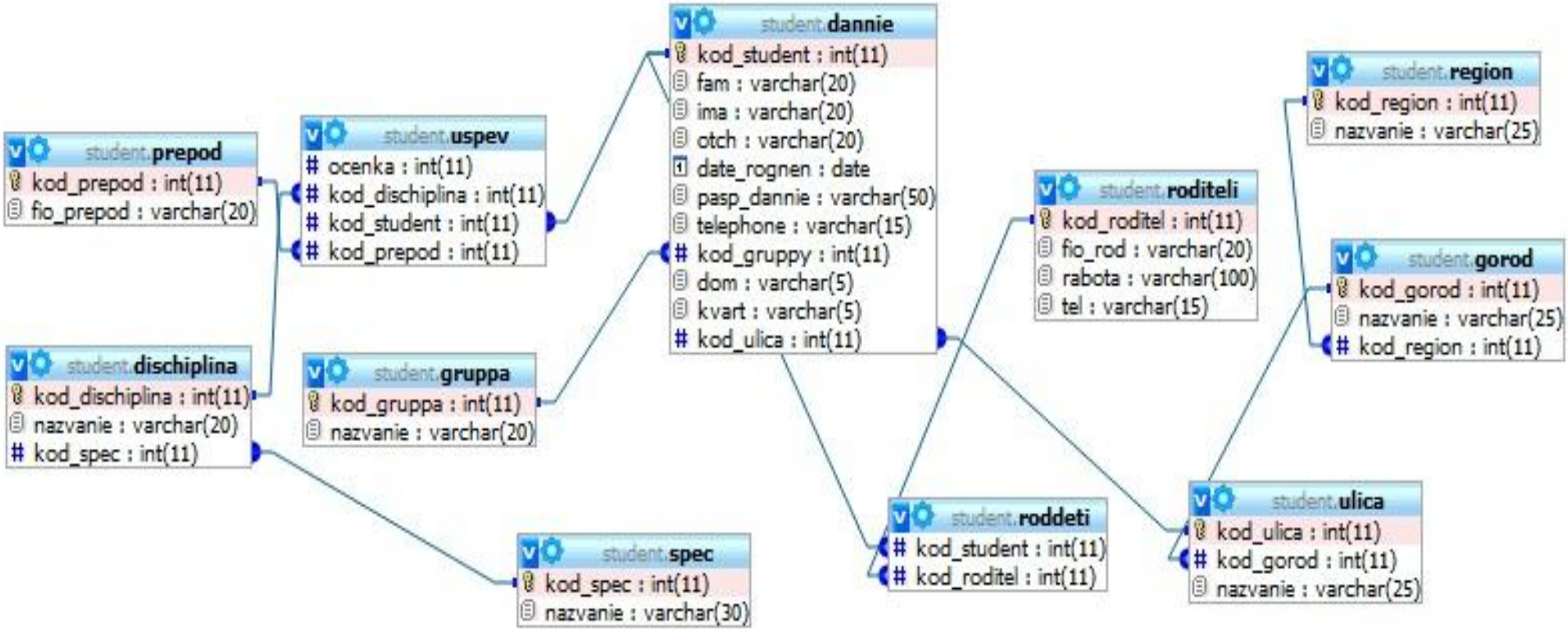

Рис. 18 Схема базы данных Student

Приведем краткое описание таблиц базы данных:

dannie – информация о студентах; **region** – регионы проживания студентов; **gorod** – города проживания студентов; **ulica** – улицы проживания студентов; **gruppa** – учебные группы; **spec** – перечень специальностей; **roditeli** – информация о родителях студентов; **roddeti** – таблица, связывающая таблицы родителей и студентов; **prepod** – информация о преподавателях; **dischiplina** – изучаемые дисциплины; **uspev** – сведения об оценках, выставленных преподавателями студентам.

Лабораторная работа №2

Выборка данных - оператор SELECT

Цель работы: Познакомить с простейшими запросами, научить выводить поля из одной таблицы, из различных таблиц.
Ключевые слова: оператор SELECT, функции DISTINCT, ALL, команда LIMIT.
Теоретический материал:

Содержимое в таблицах в SQL просматривается с помощью оператора *SELECT*. Синтаксис его использования следующий:

```
SELECT <поля> FROM <таблица>
```

Вместо <поля> необходимо указать либо имя столбца, значения которого нужно просмотреть, либо имена нескольких столбцов через запятую, либо символ звездочки «*», означающий выбор всех столбцов таблицы.

Вместо <таблица> следует указать имя таблицы.

Пример 1. Просмотреть все столбцы из таблицы RODITELI.

```
SELECT * FROM RODITELI
```

Пример 2. Вывести фамилии родителей.

```
SELECT FIO_ROD FROM RODITELI
```

Пример 3. Вывести фамилии родителей, телефоны и место работы.

```
SELECT FIO_ROD, TEL, RABOTA FROM RODITELI
```

Для вывода полей из разных таблиц используются составные имена в виде Имя_таблицы.Имя_поля.

Пример 4. Вывести названия районов и городов.

Можно вывести данные двумя запросами:

```
SELECT NAZVANIE  FROM  REGION
SELECT NAZVANIE  FROM  GOROD
```

Либо использовать составные имена:

```
SELECT REGION.NAZVANIE, GOROD.NAZVANIE FROM  REGION, GOROD
```

Для переименования выводимого поля используется конструкция **AS нов_имя_поля**, которая называется псевдонимом.

Пример 5. Вывести фамилии родителей, переименовав поле FIO_ROD в ФАМИЛИЯ.

```
SELECT FIO_ROD AS Фамилия FROM RODITELI
```

Для исключения повторяющихся записей используется функция **DISTINCT** (отличающиеся), для вывода всех записей используется функция **ALL** (все). Функция ALL используется по умолчанию.

Пример 6. Вывести должностей родителей.

```
SELECT DISTINCT (RABOTA) FROM RODITELI
```

Для вывода заданного количество строк и указания позиции используется команда **LIMIT номер_позиции, количество_строк**.

Пример 7. Вывести с 2 по 4 строки таблицы DANNIE.

```
SELECT * FROM  RODITELI  LIMIT 1,3
```

Задания:

1) Вывести данные из таблицы DANNIE.
2) Вывести данные из таблицы DISCHIPLINA.
3) Вывести фамилии всех студентов.
4) Вывести названия всех групп.
5) Вывести фамилии, имена, телефоны, паспортные данные студентов.
6) Вывести фамилии родителей и телефоны.
7) Вывести названия городов, названия улиц.
8) Вывести названия предметов и фамилии преподавателей.
9) Вывести фамилии и дату рождения студентов, переименовав поле DATE_ROGNEN в ДЕНЬ_РОЖДЕНИЯ.
10) Вывести названия улиц, переименовав поле NAZVANIE в УЛИЦЫ.
11) Вывести список улиц, исключив повторяющиеся значения.
12) Вывести различные имена студентов.
13) Вывести первую в списке специальность.
14) Вывести с 6 по 10 строки таблицы RODITELI.

Лабораторная работа №3

Уточнения запросов

Цель работы: Научиться создавать запросы с условием, в т.ч. с использованием логических операторов, сортировать данные по возрастанию и убыванию.
Ключевые слова: операторы WHERE, GROUP BY, ORDER BY, HAVING ,IN, BETWEEN, LIKE, is NULL.
Теоретический материал:

Использование в операторе SELECT предложения, определяемого ключевым словом WHERE (где), позволяет задавать выражение условия (предикат), принимающее значение истина или ложь для значений полей строк таблиц, к которым обращается оператор SELECT. Предложение WHERE определяет, какие строки указанных таблиц должны быть выбраны. В таблицу, являющуюся результатом запроса, включаются только те строки, для которых условие (предикат), указанное в предложении WHERE, принимает значение истина.

В задаваемых в предложении WHERE условиях могут использоваться операции сравнения, определяемые операторами = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно), а также логические операторы AND, OR и NOT.

Пример 1. Выбрать студентов 3 группы.
SELECT * FROM DANNIE WHERE KOD_GRUPPY=3

Пример 2. Вывести родителей, работающих водителями и поварами.
SELECT FIO_ROD FROM RODITELI WHERE RABOTA=’ВОДИТЕЛЬ’ OR RABOTA=’ПОВАР’

Оператор **GROUP BY** (группировать по) служит для группировки записей по значениям одного или нескольких столбцов. Он собирает записи в группы и упорядочивает группы по алфавиту (точнее по ASCII- кодам символов).

Пример 3. Вывести родителей студентов сгруппированных по месту работы.
SELECT RABOTA, FIO_ROD FROM RODITELI GROUP BY RABOTA

Оператор **HAVING** (имеющие, при условии) обычно применяются совместно с оператором группировки GROUP BY и задает фильтр записей в группах

Пример 4. Вывести родителей студентов сгруппированных по месту работы, если они водители или учителя.
SELECT RABOTA, FIO_ROD FROM RODITELI GROUP BY RABOTA HAVING (RABOTA=‘ВОДИТЕЛЬ’) OR (RABOTA=’УЧИТЕЛЬ’)

При задании логического условия в предложении WHERE могут быть использованы операторы IN, BETWEEN, LIKE, IS NULL.

Операторы **IN (NOT IN)** используется для сравнения проверяемого значения поля с заданным списком. Список значений указывается в скобках справа от оператора IN.

Построенное таким образом условие выполняется, если значение поля совпадает с одним из значений, перечисленных в списке. Необходимо точное совпадение. Символьный тип данных, тип данных - дата водятся в апострофах, числовой без апострофов.

Конструкция NOT IN выполняется если значение поля, имя которого указано слева от NOT IN, *не совпадает* ни с одним из значений, перечисленных в списке.

Пример 5. Вывести родителей, работающих водителями и поварами.

SELECT FIO_ROD FROM RODITELI WHERE RABOTA IN (‘ВОДИТЕЛЬ’, ’ПОВАР’)

Пример 6. Вывести всех родителей, кроме работающих учителями и врачами.

SELECT FIO_ROD FROM RODITELI WHERE RABOTA NOT IN (‘УЧИТЕЛЬ’, ’ВРАЧ’)

Оператор **BETWEEN** используется для проверки условия вхождения значения поля в заданный интервал, то есть вместо списка значений атрибута этот оператор задает границы его изменения.

Пример 7. Вывести информацию о родителях с кодом от 2 до 9:

SELECT * FROM RODITELI WHERE KOD_RODITEL BETWEEN 2 AND 10;

Левая граница (в данном случае 2) входит во множество значений, с которыми производится сравнение, правая граница (в данном случае 10) не входит. Оператор BETWEEN может использоваться как для числовых, так и для символьных типов полей.

Пример 8. Вывести фамилии родителей, заглавные буквы которых находятся в диапазоне от А до И.

SELECT * FROM RODITELI WHERE FAM BETWEEN ‘A’ AND ‘K’;

Оператор **LIKE** проверяет строковые значения полей на предмет совпадения строки целиком или её части с искомым условием.

Для выборки строковых значений по заданному образцу подстроки можно применять шаблон искомого образца строки, использующий следующие символы:

• символ подчеркивания «_», указанный в шаблоне, определяет возможность наличия в указанном месте одного любого символа;

• символ «%» допускает присутствие в указанном месте проверяемой строки последовательности любых символов произвольной длины.

Этот оператор применим только к символьным полям типа CHAR или VARCHAR.

Пример 9. Выбрать сведения о родителях, фамилии которых начинаются на букву «П».

SELECT * FROM RODITELI WHERE FAM LIKE 'П%'

Оператор **IS NULL** выводит строки, содержащие значение NULL в проверяемом поле.

Пример 10. Выбрать записи, содержащие пустые значения в данных о работе родителей.

SELECT * FROM RODITELI WHERE RABOTA IS NULL;

Для сортировки в SQL существует ключевое слово ***ORDER BY,*** после которого указывается имя столбца, по которому будет происходить сортировка. Команда имеет следующий синтаксис:

```
SELECT <поля> FROM <таблица> ORDER BY <поле>
```

После ORDER BY <поле> можно указать направление сортировки. По умолчанию записи сортируются по возрастанию (***ASC***), ключевое слово ***DESC*** задает сортировку в порядке, обратном алфавитному.

Пример 11. Отсортировать фамилии родителей в алфавитном порядке.

SELECT FIO_ROD FROM RODITELI ORDER BY FIO_ROD

Пример 12. Отсортировать место работы родителей в порядке, обратном алфавитному.

SELECT RABOTA FROM RODITELI ORDER BY RABOTA DESC

Сортировку можно производить сразу по нескольким столбцам. В этом случае сортировка идет сначала по первому полю в указанном направлении, а если в этом столбце есть одинаковые значения, то они будут отсортированы по второму полю.

Пример 13. Отсортировать место работы родителей в алфавитном порядке, а фамилии в порядке, обратном алфавитному.

SELECT FIO_ROD , RABOTA FROM RODITELI ORDER BY FIO_ROD, FIO_ROD DESC;

Задания:

1. Вывести фамилии студентов, обучающихся в группе с кодом 3.
2. Вывести название региона с кодом 1.
3. Вывести фамилию преподавателя с кодом 2.
4. Вывести информацию о студентах, обучающихся в группах с кодами 1 и 2.
5. Вывести названия дисциплин с кодами 2 и 3.

6. Вывести имена студентов, заглавные буквы которых находятся в диапазоне от «В» до «М».

7. Вывести данные о студентах, фамилии которых начинаются на букву «М».

8. Выбрать записи, содержащие пустые значения о номере квартире студентов (проживающих в доме).

9. Отсортировать фамилии студентов в алфавитном порядке.

10. Вывести данные о студентах, отсортировав номера телефонов по возрастанию.

11. Отсортировать студентов по номеру группы в порядке возрастания, а для одинаковых групп - фамилии в порядке, обратном алфавитному.

12. Выбрать студентов, фамилии которых заканчиваются на «а».

13. Отсортировать в порядке возрастания фамилии студентов, чьи отчества заканчиваются на «ич».

14. Отсортировать в порядке возрастания фамилии студентов, обучающихся в группе 3.

15. Вывести студентов, родившихся в 1990 году.

16. Отсортировать в алфавитном порядке фамилии студентов, у которых начальные буквы имени заключены в диапазоне от «И» до «Я».

17. Вывести данные о студентах с фамилиями «Петров», «Смелов».

18. Вывести в порядке, обратном алфавитному, все имена и фамилии студентов, родившихся не в 1991 году.

19. Вывести информацию о всех студентах, кроме Варечкина и Климовой.

20. Выбрать студентов, у которых номера телефонов связи МТС и начинаются с 3.

21. Сгруппировать список улиц по коду города.

22. Сгруппировать данные о студентах по дате рождения.

23. Вывести по группам коды городов для улиц Ставропольская и Комсомольская.

24. Сгруппировать по улицам данные о студентах, обучающихся в группе с кодом 1.

25. Отсортировать в алфавитном порядке фамилии студентов, у которых в отчестве встречается «..ев..».

Лабораторная работа №4

Агрегированные функции

Цель работы: Научиться находить сумму полей, максимальное, минимальное, среднее значения полей, подсчитывать количество записей.

Ключевые слова: функции COUNT, SUM, MAX, MIN, AVG.

Теоретический материал:

Агрегированные функции используются подобно именам полей в предложении SELECT запроса, но с учетом того, что они берут имена полей в качестве аргумента. С SUM и AVG используются только числовые поля, а с COUNT, MAX, MIN могут использоваться числовые или символьные поля.

Рекомендуем выполнить примеры, с созданной ранее базой данных и посмотреть результаты.

Функция **COUNT** производит подсчет количества строк или не-NULL значений полей, которые выбрал запрос.

Пример 1. Подсчитать количество записей в таблице DANNIE.

```
SELECT COUNT(*) FROM  DANNIE
```

В результате выполнения этого запроса появится столбец с заголовком COUNT(*), поэтому можно использовать оператор переименования.

Пример 2. Подсчитать количество записей в таблице DANNIE и назвать поле КОЛИЧЕСТВО.

```
SELECT COUNT(*)  AS КОЛИЧЕСТВО FROM  DANNIE
```

Функция **SUM** рассчитывает арифметическую сумму всех выбранных значений данного поля.

Пример 3. Вывести сумму оценок студентов сгруппированных по номеру группы.

```
SELECT  KOD_GRUPPY, SUM(OCENKA) FROM DANNIE, USPEV  GROUP BY KOD_GRUPPY
```

Функция **AVG** – производит усреднение всех выбранных значений данного поля.

Пример 4. Вывести среднее значения оценок.

```
SELECT AVG(OCENKA) FROM  USPEV
```

Функция **MAX** – находит и возвращает наибольшее из всех выбранных значений данного поля.

Пример 5. Вывести максимальную оценку.

```
SELECT MAX(OCENKA) FROM  USPEV
```

Функция **MIN** – находит и возвращает наименьшее из всех выбранных значений данного поля.

Пример 6. Вывести минимальную оценку студентов.

SELECT MIN(OCENKA) FROM USPEV

Ключевое слово **GROUP BY** – указывает условие группировки строк.

Пример 7. Вывести среднее оценок, максимальную оценку, минимальную оценку студентов сгруппированных по номеру группы, с указанием имени у каждого столбца.

SELECT KOD_STUDENT, AVG(OCENKA) AS СРЕДНЯЯ, MAX(OCENKA) AS МАКСИМАЛЬНАЯ, MIN (OCENKA) AS МИНИМАЛЬНАЯ FROM USPEV GROUP BY KOD_STUDENT

Пример 8. Вывести коды и численность групп, в которых более 2 человек.

SELECT KOD_GRUPPY, COUNT(*) FROM DANNIE GROUP BY KOD_GRUPPY HAVING COUNT(*)>2

Задания:

Внимание: необходимо переименовать каждое вычисляемое поле.

1. Найти среднее значение оценок по каждому студенту.
2. Найти максимальную оценку по каждой дисциплине.
3. Найти среднюю оценку, выставленную каждым преподавателем.
4. Вывести минимальную оценку, выставленную каждым преподавателем.
5. Перевести каждую оценку в рейтинговый бал (оценка, большая 3 баллов, увеличивается в 2 раза).
6. Подсчитать количество разных групп.
7. Подсчитать количество различных квартир.
8. Вывести среднюю оценку, максимальную оценку, минимальную оценку для студента с кодом 3.
9. Подсчитать количество хороших оценок.
10. Подсчитать процент двоек, выставленных каждым преподавателем.
11. Посчитать количество и сумму 5-к и 4-к.
12. Подсчитать процент качества и процент успеваемости (общее количество оценок 26).
13. На скольких улицах проживают более 1 студента.
14. Вывести количество оценок, для которых выполняется условие «оценка*2+1>10».

Лабораторная работа №5

Строковые и числовые функции

Цель работы: Познакомиться с основными строковыми и числовыми функциями.
Ключевые слова: функции UCASE, UPPER, LCASE, LOWER, MID, LEN, CONCAT
Теоретический материал:

Рекомендуем выполнить примеры, с созданной ранее базой данных и посмотреть результаты.

Основные строковые функции

Функция **UCASE** преобразует символы в верхний регистр.

Пример 1. Вывести фамилии студентов заглавными буквами, переименовав поле fam в familio.

```
SELECT UCASE(FAM) AS FAMILIO  FROM DANNIE
```

Функция **UPPER** – переводит все символы указанной в параметре строки в верхний регистр (работает только с латиницей).

Функция **LCASE** преобразует символы в нижний регистр.

Пример 2. Вывести фамилии студентов строчными буквами, переименовав поле fam в familio.

```
SELECT LCASE(FAM) AS FAMILIO  FROM DANNIE
```

Функция **LOWER** – переводит все символы указанной в параметре строки в нижний регистр (работает только с латиницей).

Функция **CONCAT(str1,str2...)** возвращает строку, созданную путем объединения аргументов (аргументы указываются в скобках - str1,str2...)., аргументами являются имена полей.

Пример 3. Вывести фамилию и имя студента в одном поле.

```
SELECT CONCAT(FAM, IMA)  FROM  DANNIE
```

Результатом будет строка, состоящая из фамилии и имени, не разделенных пробелом. Для добавления пробела запрос нужно изменить:

```
SELECT concat(fam,'  ', ima)  FROM `dannie`
```

Функция **INSERT(str, pos, len, new_str)** возвращает строку str, в которой подстрока, начинающаяся с позиции pos и имеющая длину len символов, заменена подстрокой new_str.

Пример 4. Вывести фамилии студентов с 3 символа (вставить с 1 позиции 3 пробела).

```
SELECT INSERT(FAM, 1, 3, ' ') FROM DANNIE
```

Функция **LENGTH(str)** возвращает длину строки str.

Пример 5. Вывести фамилия студента и количество символов в ней.
SELECT FAM, LENGTH(FAM) FROM DANNIE

Функция **REPEAT(str, n)** возвращает строку str n-количество раз.

Пример 6. Вывести фамилию студента 3 раза в одном поле.
SELECT REPEAT(`fam`,3) FROM `dannie`

Функция **REPLACE(str, pod_str1, pod_str2)** возвращает строку str, в которой все подстроки pod_str1 заменены подстроками pod_str2.

Пример 7. В названиях городов заменить длинное 'Армавир' на короткое 'Ар'.
SELECT REPLACE(NAZVANIE,'АРМАВИР','АР') FROM GOROD

Функция **REVERSE(str)** возвращает строку str, записанную в обратном порядке.

Пример 8. Написать фамилии студентов в обратном порядке.
SELECT REVERSE(FAM) FROM DANNIE

Основные числовые функции

Функция **POSITION('подстрока' IN 'строка')** – ищет вхождение подстроки в строке. В случае успешного поиска возвращает номер положения ее первого символа. Иначе – 0. Если подстрока имеет нулевую длину, то функция возвращает 1. Если хотя бы один из параметров имеет значение NULL, то возвращает NULL. Нумерация ведется слева направо, начиная с 1.

Пример 9. Определить позицию буквы «Е» в строке «ПРИВЕТ ВСЕМ»
SELECT POSITION('Е' IN 'ПРИВЕТ ВСЕМ')

Пример 10. Вывести данные о студентах, родившихся в 1991 году.
SELECT * FROM DANNIE WHERE POSITION('1991' IN DATE_ROGNEN)>0

Функция **ABS(число)** возвращает абсолютное значение числа.

Пример 11. Найти |-20|.
SELECT abs(-20)

Функция **MOD(число1, число2)** возвращает остаток от целочисленного деления первого числа на второе.

Пример 12. Найти остаток от деления 8 на 3
SELECT MOD(8,3)

Функция **SQRT(число)** возвращает арифметический квадратный корень из числа.

Пример 13. Вычислить $\sqrt{169}$

```
SELECT SQRT(169)
```

Функция **FLOOR(число)** округляет число в большую сторону. Функция **CELL(число)** округляет число в меньшую сторону.

Задания:

1. Вывести фамилии родителей заглавными буквами.
2. Вывести название улиц маленькими буквами.
3. Вывести названия факультетов, курс, группу (вырезать из полного названия группы название факультета, например, из МФ-МАТ-4-1 должно получиться МФ-4-1).
4. Вывести данные о родителях, разместив информацию о работе и телефоне в одном поле.
5. Вывести имя, отчество, телефон студента и количество символов в них.
6. Вывести номер телефона в обратном порядке.
7. Вывести дату рождения 5 раз в одном поле.
8. Заменить 1991 год рождения на 91.
9. Вывести количество студентов, у которых серия в паспортных данных 03 01.
10. Найти |38-20-168|, используя числовые функции.
11. Найти остаток от деления 16 на 5.
12. Вычислить $\sqrt{22500}$.
13. Округлить число в меньшую сторону 5,128, округлить число в большую сторону 5,265.

Лабораторная работа №6

Условные выражения с оператором CASE

Цель работы: научится использовать оператор условного перехода CASE.
Ключевые слова: оператор CASE со значениями, оператор CASE с условиями поиска.

Рекомендуем выполнить примеры, с созданной ранее базой данных и посмотреть результаты.

Теоретический материал:

В обычных языках программирования имеются операторы условного перехода, которые позволяют управлять вычислительным процессом в зависимости от того, выполняется или нет некоторое условие. В языке SQL таким оператором является CASE (случай, обстоятельство, экземпляр). Он имеет две основные формы.

Оператор CASE со значением имеет следующий синтаксис:

CASE *проверяемое_значение*
WHEN *значение1* THEN *результат1*
WHEN *значение2* THEN *результат2*
….
WHEN *значениеN* THEN *результатN*
ELSE *результатX*
END

В случае, когда *проверяемое_значение* равно *значение1*, оператор CASE возвращает значение *результат1*. В противном случае *проверяемое_значение* сравнивается с *значение2*, и если они равны, возвращается *результат2* и.т.д. Если *проверяемое_значение* не равно ни одному из таких значений, то возвращается значение *результатX*.

Ключевое слово ELSE не является обязательным. Если оно отсутствует и ни одно из значений, подлежащих сравнению, не равно проверяемому значению, то возвращается значение NULL.

Пример 1. Вывести название региона и код региона (Краснодарский край – 93, Ставропольский край – 73)

```
SELECT `nazvanie` ,
CASE `nazvanie`
WHEN 'Краснодарский край' THEN '93'
WHEN 'Ставропольский край' THEN '73'
ELSE `nazvanie`
END
```

AS Код_региона
FROM `region`

Вторая форма оператора CASE предполагает его использование при поиске в таблице тех записей, которые удовлетворяют определенному условию:

```
CASE
      WHEN условие1 THEN результат1
      WHEN условие2 THEN результат2
      ….
      WHEN условиеN THEN результатN
ELSE результатX
END
```

Оператор CASE проверяет, истинно ли *условие1* для первой записи в наборе, определенном оператором WHERE, или во всей таблице, если оператор WHERE отсутствует. Если да, то CASE возвращает значение *результат1*. В противном случае для данной записи проверяется *условие2* и т.д. Если ни одно из условий не выполняется, то возвращается значение *результатX*, указанное после ключевого слова ELSE.

Ключевое слово ELSE не является обязательным. Если оно отсутствует и ни одно из значений, подлежащих сравнению, не равно проверяемому значению, то возвращается значение NULL.

Пример 2. Вывести фамилии, имена студентов и номера телефонов. Вместо значения NULL вывести фразу «Нет телефона»

```
SELECT `fam`, `ima`,
CASE WHEN `telephone` IS NULL THEN 'НЕТ ТЕЛЕФОНА'
ELSE `telephone`
END
AS Номер_телефона
FROM `dannie`
```

Обратите внимание, что вместо первой формы оператора CASE всегда можно использовать вторую.

```
CASE
      WHEN проверяемое_значение=значение1 THEN результат1
      WHEN проверяемое_значение=значение2 THEN результат2
      ….
      WHEN проверяемое_значение=значениеN  THEN результатN
ELSE результатX
      END
```

Пример 3. Вывести ФИО родителей и номера их телефонов с указанием мобильного оператора (8918…, 8919… - МТС, 8928… - МЕГАФОН, 8905…, 8906…, 8909… – БИЛАЙН)

```
SELECT `fio_rod`,`tel`,
CASE
WHEN (`tel` like '8918%')or (`tel` like '8919%')  THEN 'МТС'
WHEN `tel` like '8928%'  THEN 'МЕГАФОН'
WHEN (`tel` like '8905%')or (`tel` like '8906%')  THEN 'БИЛАЙН'
ELSE `tel`
END
AS Оператор
FROM `roditeli`
```

Порядок выполнения работы:

1. Перевести каждую оценку в рейтинговый бал (за оценку меньше 3 начисляется 0 баллов, от 3 до 4 – 1 балл, за оценку 5 – 2 балла).
2. Вывести список оценок и их буквенное обозначение (5 – «отлично», 4 – «хорошо», 3 – «удовлетворительно», 2 – «неудовлетворительно»).
3. Вывести список оценок и указать значение по системе «зачет-незачет» (для оценок 5 или 4 – «зачет», для остальных – «незачет»).
4. Вывести названия групп и названия специальностей («…ПИЭ…» - Прикладная информатика в экономике, «…Мат…» - Математика, «…Инф…» - Информатика, в случае другого обозначения повторить название группы).
5. Вывести фамилии студентов и место прохождения практики (студенты группы с кодом 1 проходят практику в «Банк УралСиб», с кодом 2 – «СберБанк», с кодом 3 – «Первомайский», с кодом 4 – «РосСельхозБанк»).

Лабораторная работа №7

Простые подзапросы

Цель работы: Научиться создавать простые подзапросы.
Ключевые слова: простые подзапросы.
Теоретический материал:

Рекомендуем выполнить примеры, с созданной ранее базой данных и посмотреть результаты.

Подзапрос – это запрос на выборку данных, вложенный в другой запрос.

SELECT <*поля*> FROM <*таблица*> WHERE (HAVING) УСЛОВИЕ (SELECT <*поля*> FROM <*таблица*> WHERE <*условие*>)

В свою очередь подзапрос может содержать другой подзапрос, но в первую очередь выполняется подзапрос, имеющий самый глубокий уровень вложения.

Часто, но не всегда, внешний запрос обращается к одной таблице, а подзапрос - к другой.

Простые подзапросы характеризуются тем, что они формально никак не связаны с содержащими их внешними запросами, что позволяет сначала выполнить подзапрос, результат которого используется для выполнения внешнего запроса.

Три вида простых подзапросов:

- подзапросы, возвращающие единственное значение;
- подзапросы, возвращающие список значений, из одного столбца таблицы;
- подзапросы, возвращающие набор записей.

Подзапросы, возвращающие единственное значение

Пример 1. Вывести оценки, которые больше среднего значения всех оценок

```
SELECT * FROM  USPEV WHERE OCENKA > (SELECT AVG(OCENKA)
FROM USPEV)
```

Подзапросы, возвращающие список значений, из одного столбца таблицы

Пример 2. Определить коды родителей студента Маркова и Иванова.

```
SELECT KOD_RODITEL FROM RODDETI WHERE KOD_STUDENT IN
(SELECT KOD_STUDENT  FROM  DANNIE WHERE FAM='МАРКОВ' OR
FAM='ИВАНОВ’)
```

Пример 3. Определить название дисциплин, которые сдавал студент с кодом 2.

```
SELECT NAZVANIE FROM DISCHIPLINA WHERE  KOD_DISCHIPLINA IN
(SELECT  KOD_DISCHIPLINA FROM  USPEV WHERE KOD_STUDENT='2')
```

Пример 4. Вывести фамилию, и место работы родителей студента с кодом 3.

SELECT FIO_ROD, RABOTA FROM RODITELI WHERE KOD_RODITEL IN (SELECT KOD_RODITEL FROM RODDETI WHERE KOD_STUDENT=3)

Порядок выполнения работы:

1. Вывести список оценок, которые получил студент Воркин.
2. Вывести все города Краснодарского края.
3. Вывести название группы студента Маркова.
4. Вывести название улицы, на которой живет студент Варечкин.
5. Определить фамилию преподавателя, поставившему студенту Климову оценку по программированию.
6. Определить специальность, на которой обучается Климова.
7. Какие оценки были получены по дисциплине Базы данных.
8. Найти дисциплины, по которым оценки ставил преподаватель Плюшкин.
9. Определить, где работают родители Смелова. Вывести полный адрес студента Петрова (регион, город, улица, дом и квартира).

Лабораторная работа №8

Связанные подзапросы

Цель работы: научится использовать связанные подзапросы.
Ключевые слова: связанные (соотнесенные / коррелированные) подзапросы.
Теоретический материал:

Рекомендуем выполнить примеры, с созданной ранее базой данных и посмотреть результаты.

Пример 1. Вывести название дисциплин, по которым получена оценка 5.

SELECT NAZVANIE FROM DISCHIPLINA WHERE 5 IN (SELECT OCENKA FROM USPEV WHERE KOD_DISCHIPLINA= DISCHIPLINA.KOD_DISCHIPLINA)

Такой подзапрос отличается от простого подзапроса тем, что вложенный подзапрос не может быть обработан прежде, чем будет обрабатываться внешний подзапрос. Это связано с тем, что вложенный подзапрос зависит от значения DISCHIPLINA.KOD_DISCHIPLINA, а оно изменяется по мере того, как система проверяет различные строки таблицы DISCHIPLINA. Следовательно, с концептуальной точки зрения обработка осуществляется следующим образом:

1. Система проверяет первую строку таблицы DISCHIPLINA. Предположим, что это строка дисциплины с номером 1. Тогда значение DISCHIPLINA.KOD_DISCHIPLINA будет в данный момент имеет значение, равное 1, и система обрабатывает внутренний запрос:

(SELECT OCENKA FROM USPEV WHERE KOD_DISCHIPLINA= 1)

получая в результате множество (1, 4, 3, 5, 2, 4, 3, 3, 5). Теперь система может завершить обработку для дисциплины с номером 1. Выборка значения NAZVANIE для KOD_DISCHIPLINA= 1 (База данных) будет проведена тогда и только тогда, когда OCENKA=5 будет принадлежать этому множеству, что, очевидно, справедливо.

2. Далее система будет повторять обработку такого рода для следующей дисциплины и т.д. до тех пор, пока не будут рассмотрены все строки таблицы DISCHIPLINA.

Подобные подзапросы называются *связанными / соотнесенными / коррелированными*, так как их результат зависит от значений, определенных во внешнем подзапросе. Обработка связанного подзапроса, следовательно, должна повторяться для каждого значения извлекаемого из внешнего подзапроса, а не выполняться раз и навсегда.

Порядок выполнения работы:

1. Вывести фамилии преподавателей, которые поставили хотя бы одну двойку.
2. Вывести название предметов, средняя оценка по которым выше 3.
3. Вывести фамилии студентов, у которых имеются оценки 3 и 4 (одновременно).
4. Вывести фамилии студентов, которые получили хотя бы одну оценку, выше средней.
5. Вывести названия групп, в которых обучается 6 студентов.

Лабораторная работа №9

Операции соединения

Цель работы: Научиться соединять таблицы различными способами.
Ключевые слова: естественное соединение, условное соединение.
Теоретический материал:

Внутренние соединения таблиц

Естественное соединение (NATURAL JOIN)

Декартовым произведением двух таблиц называется таблица, составленная из всевозможных сочетаний записей обеих таблиц.

Пример 1. Вывести декартово произведение таблиц DANNIE и USPEV.

```
SELECT * FROM  DANNIE, USPEV
```

Общим столбцом этих таблиц является столбец kod_student.

В полученном декартовом произведении интересуют не все данные, а только те, в которых идентичные столбцы имеют одинаковые значения. Кроме того, в результативной таблице не нужны два идентичных столбца, достаточно лишь одного из них.

```
SELECT DANNIE.*, USPEV.OCENKA FROM  DANNIE, USPEV WHERE
DANNIE.KOD_STUDENT=USPEV.KOD_STUDENT
```

В результате выполнения этого запроса получается таблица естественным соединением.

Данный запрос можно записать, используя псевдонимы.

```
SELECT T1.*, T2.OCENKA FROM  DANNIE T1, USPEV T2 WHERE
T1.KOD_STUDENT=T2.KOD_STUDENT
```

Эквивалентный запрос можно составить с помощью оператора **NATURAL JOIN**

```
SELECT T1.*, T2.OCENKA FROM  DANNIE T1 NATURAL JOIN USPEV T2.
```

При естественном соединении, выполняемом с помощью NATURAL JOIN, проверяется равенство всех одноимённых столбцов соединяемых таблиц.

Условное соединение (JOIN …ON)

Условное соединение похоже на соединение с условием равенства. В качестве условия может выступать любое логическое выражение, которое записывается после ключевого слова ON (при), а не WHERE. Если условие выполняется для текущей записи декартового произведения, то она входит в результативную таблицу.

Пример 2. Вывести информацию о студентах и их оценках по дисциплине с кодом 2.

```
SELECT * FROM  DANNIE  JOIN USPEV ON (DANNIE.KOD_STUDENT =
USPEV.KOD_STUDENT) AND (USPEV.KOD_DISCHIPLINA=2)
```

Соединение по именам столбцов (JOIN …USING)

Соединение по именам столбцов похоже на естественное соединение. Отличие состоит в том, что можно указать, какие именно одноименные столбцы должны проверяться, а при естественном проверяются все одноименные столбцы.

Пример 3. Вывести в каком городе какие улицы.

SELECT * FROM GOROD JOIN ULICA USING (KOD_GOROD)

Данные таблицы содержат более одного идентичного поля, поэтому естественное соединение вернет пустой результат.

Пример 4. Запрос можно сформулировать иначе:

SELECT * FROM GOROD INNER JOIN ULICA ON (GOROD.KOD_GOROD= ULICA. KOD_GOROD)

Внешние соединения таблиц

Из таблицы, получаемой при внутреннем соединении, отбраковываются все записи, для которых нет соответствующей записи одновременно в обеих таблицах. При внешнем соединении такие несоответствующие записи сохраняются.

Левое соединение (LEFT OUTER JOIN)

При левом внешнем соединении несоответствующие записи, имеющиеся в левой таблице, сохраняются в результативной, а имеющиеся в правой – удаляются.

Рассмотренный в ***Примере 1*** запрос выводит не все записи, тот же самый запрос при внешнем левом объединении выглядит так:

SELECT T1.*, T2.OCENKA FROM DANNIE T1 LEFT OUTER JOIN USPEV T2 ON T1.KOD_STUDENT=T2.KOD_STUDENT

Правое соединение (RIGHT OUTER JOIN)

При правом внешнем соединении несоответствующие записи, имеющиеся в правой таблице, сохраняются в результативной, а имеющиеся в левой – удаляются.

Рассмотренный в ***Примере 1*** запрос выводит не все записи, тот же самый запрос при внешнем правом объединении выглядит так:

SELECT T1.*, T2.OCENKA FROM DANNIE T1 RIGHT OUTER JOIN USPEV T2 ON T1.KOD_STUDENT=T2.KOD_STUDENT

Полное соединение (FULL JOIN)

Полное соединение выполняет и левое, и правое внешние соединения.

SELECT * FROM DANNIE FULL JOIN USPEV

Задания:

1. Вывести названия регионов и соответствующие названия городов.
2. Вывести перечень специальностей и название групп.
3. Вывести названия дисциплин, по которым студенты получили 5.
4. Вывести фамилии преподавателей, поставивших 3.
5. Вывести фамилии студентов и названия соответствующих улиц.
6. Вывести фамилии студентов и названия соответствующих городов.
7. Вывести фамилии студентов и названия соответствующих регионов.
8. Вывести информацию о студентах и их родителях.

Лабораторная работа №10

Добавление, удаление и изменение данных

Цель работы: Научиться создавать запросы на добавление, удаление и изменение данных в таблицах.

Ключевые слова: INSERT, DELETE, UPDATE.

Теоретический материал:

Добавление новых записей

Для добавления записи в таблицу служит оператор **INSERT**, который имеет несколько форм:

INSERT INTO <*таблица*> VALUES (<*список значений*>)

– вставляет пустую запись в указанную таблицу и заполняет эту запись значениями из списка, указанного за ключевым словом VALUES. При этом первое в списке значение вводится в первый столбец, второе – во второй и т.д. Порядок столбцов задается при создании таблицы. Данная форма не очень надежна, поскольку нетрудно ошибиться в порядке вводимых значений.

INSERT INTO <*таблица*> (<*список столбцов*>) VALUES (<*список значений*>)

– вставляет пустую запись в указанную таблицу и в заданные столбцы значения из указанного списка. При этом в первый столбец из *список столбцов* вводится первое значение из *список значений.* Порядок и количество имен столбцов в списке может отличаться от их порядка и количества, заданного при создании таблицы. Столбцы которые не указаны в списке, заполняются значением NULL.

Пример 1. Добавить в DANNIE Иванова Ивана Ивановича.

INSERT INTO DANNIE (FAM, IMA, OTCH) VALUES ('ИВАНОВ', 'ИВАН', 'ИВАНОВИЧ')

Возможна работа со значениями типа запись. Это позволяет за ключевым словом VALUES указывать несколько наборов значений в круглых скобках.

Пример 2. Добавить в таблицу Города записи НОВОКУБАНСК, КУРГАНИНСК, КРОПОТКИН.

INSERT INTO GOROD (NAZVANIE) VALUES ('НОВОКУБАНСК'), ('КУРГАНИНСК'), ('КРОПОТКИН')

INSERT INTO <*таблица*> (<*список столбцов*>) <*запрос на выборку*>

– вставляет в указанную таблицу записи, возвращаемые запросом на выборку.

Пример 3. Названия городов добавить в названия регионов.

INSERT INTO REGION(NAZVANIE) SELECT NAZVANIE FROM GOROD

Удаление записей

Для удаления записей из таблицы применяют запрос:

DELETE FROM <*таблица*> WHERE <*условие*>

Данный оператор удаляет из указанной таблицы записи (а не отдельные значения полей), которые удовлетворяют указанному условию.

В операторе WHERE может находиться подзапрос на выборку данных.

Операция удаления записей является необратимой, чтобы избежать удаления нужных данных рекомендуется сначала выполнять запросы, чтобы просмотреть какие записи будут удалены.

Пример 4. Удалить студентов, у которых сумма оценок меньше 10.

Рекомендуется создать запрос на выборку таких студентов

SELECT * FROM DANNIE T1 WHERE 10 < (SELECT SUM(T2.OCENKA) FROM USPEV T2 WHERE T1.KOD_STUDENT=T2.KOD_STUDENT)

Запрос на удаление выглядит следующим образом:

DELETE FROM DANNIE T1 WHERE 10 < (SELECT SUM(T2.OCENKA) FROM USPEV T2 WHERE T1.KOD_STUDENT=T2.KOD_STUDENT)

Для удаления всех записей из таблицы достаточно использовать:

DELETE FROM <*таблица*>

При этом сама таблица остается и готова для вставки новых записей.

Изменение данных

Для изменения значений столбцов применяется команда:

UPDATE <*таблица*> SET <*поле*> = <*значение*> WHERE <*условие*>

Пример 5. Изменить место работы Власовой Валентины Васильевны – продавец на секретарь.

UPDATE RODITELI SET RABOTA=’СЕКРЕТАРЬ’ WHERE FIO_ROD=’Власова Валентина Васильевна’

Порядок выполнения работы:

1. Добавить в таблицу о студентах одного студента.
2. Добавить в таблицу о родителях информацию о двух родителях студента.
3. Названия улиц добавить в названия городов.
4. Добавить названия дисциплин в названия специальностей.
5. Удалить из таблиц Города, названия улиц.
6. Удалить из таблицы Специальности названия дисциплин.
7. Удалить информацию о добавленном студенте.
8. Удалить информацию о родителях добавленного студента.
9. Изменить номера телефонов МТС на Мегафон (8918… на 8928…).
10. Изменить название дисциплины ‘Информатика’ на ‘Информатика и ИТ’.
11. Воркин Фома Григорьевич перевелся в группу с кодом 2. Внесите соответствующие изменения в таблицу DANNIE.

Лабораторная работа №11

Представления

Цель работы: познакомиться с понятием представление, научиться создавать представления, изменять данные в представлениях.

Ключевые слова: представления.

Теоретический материал:

Представления – это виртуальные таблицы, но они могут быть доступны многим пользователям и существуют в базе данных до тех пор, пока не будут принудительно удалены. Они во всем похожи на обычные таблицы базы данных, за исключением того, что не являются физическими объектами хранения данных.

Данные в представлениях выбираются из таблиц, т.е. представляются в том или ином виде. Они применяются, чтобы скрыть от пользователя некоторые столбцы, скомбинировать из нескольких таблиц одну, которая часто нужна пользователю, а запрос для неё очень сложен. Таким образом, представления используются как надстроечные средства для адаптации базы данных к различным категориям пользователей.

Представления создаются с помощью оператора ***CREATE VIEW*** (создать вид, представление).

CREATE VIEW <*имя представления*> AS <*запрос*>

Пример 1. Создать представление, которое выводит фамилии и соответствующие оценки студентов.

CREATE VIEW OCENKI AS

SELECT DANNIE.FAM, USPEV.OCENKA FROM DANNIE, USPEV WHERE DANNIE.KOD_STUDENT = USPEV. KOD_STUDENT.

Создана виртуальная таблица *OCENKI,* к которой можно обращаться с запросами как к обычной таблице.

Пример 2. Вывести из представления *OCENKI*, только фамилии и хорошие оценки.

*SELECT * FROM OCENKI WHERE OCENKA IN (4,5)*

Рассмотренное представление является многотабличным, поскольку создано на основе не одной, а двух таблиц. На практике используются более простые однотабличные представления, в которых скрываются некоторые столбцы и/или добавляются, значения которых вычисляются.

Пример 3. Создать представление Rod, в котором будут отображены фамилии родителей и их телефоны.

CREATE VIEW ROD AS SELECT FIO_ROD AS FIO, TEL FROM RODITELI.

Название представлений и таблиц не должны совпадать. Данное представление можно заменить обычным запросом

SELECT FIO_ROD AS FIO, TEL FROM RODITELI

Однако относительно полученного набора данных нельзя задать какой–нибудь запрос, поскольку этот набор является виртуальной таблицей, отличной от представления.

Оператор *CREATE VIEW* допускает и такую форму синтаксиса:

CREATE VIEW *<имя представления>* (*<столбец1>, <столбец2>,…, <столбецN>*) AS *<запрос>*

Пример 4. Создать представление, которое выводит фамилию и дату рождения студентов.

CREATE VIEW DATE (FIO, DATE) AS SELECT FAM, DATE_ROGNEN FROM DANNIE

В представлении указываются имена столбцов, которые могут быть отличны от имён столбцов в таблице.

Удаление представления осуществляется командой:

DROP VIEW *<имя представления>*

Порядок выполнения работы:

1. Создать представление DAN, которое выводит фамилию, паспортные данные студента, название улицы, на которой проживает студент.
2. В представлении DAN вывести отсортировать данные о студентах по улицам в алфавитном порядке.
3. В представлении DAN вывести студентов, проживающих на улицах, начинающих на букву К.
4. Создать представление MINMAX, которое выводит фамилию студента, минимальную, максимальную оценку.
5. В представлении MINMAX найти среднее минимальное и максимальное значение оценок.
6. В представлении MINMAX найти среднее минимальное и максимальное значение оценок для каждого студента.
7. Создать представление OBUCH (fio, gruppa, spec) , включающее поля фамилия студента, название группы, название специальности.
8. В представлении OBUCH подсчитать количество студентов в каждой группе.
9. Создать представление ADRES (fio, region, gorod, ulica, dom, kvart).
10. В представление ADRES вывести студентов по городам. Вывести количество студентов проживающих в разных городах.
11. В представление ADRES вывести студентов по регионам. Вывести количество студентов проживающих в разных регионах.
12. Вывести студентов проживающих в одном городе, на одной улице.

Лабораторная работа №12

Связь MYSQL и DELPHI

Цель работы: научиться обрабатывать информацию из базы данных MySQl с помощью прикладных программ на примере DELPHI.

Ключевые слова: MYSQL, DELPHI.

Задание:

В первую очередь, создаём тестовую базу данных, к которой необходимо будет подключиться. Пусть база называется primer и содержит одну таблицу zarp с полями fio (varchar(20)) и zarp (int). Заполняем ее данными, причем записи вносим как на русском, так и на латинском языке (Рис. 19):

fio	zarp
Иванов	1000
Petrov	2000

Рис. 19 Данные таблицы

Запускаем Delphi.

На форме размещаем компоненты:

- Button с вкладки Standart (Рис. 20),

Рис. 20 Вкладка Standart

- Data Sourse с вкладки Data Access (Рис. 21),

Рис. 21 Вкладка Data Access

- DBGrid с вкладки Data Controls (Рис. 22),

Рис. 22 Вкладка Data Controls

- SQLConnection, SimpleDataSet с вкладки dbExpress (Рис. 23).

Рис. 23 Вкладка dbExpress

Форма должна выглядеть следующим образом (Рис. 24):

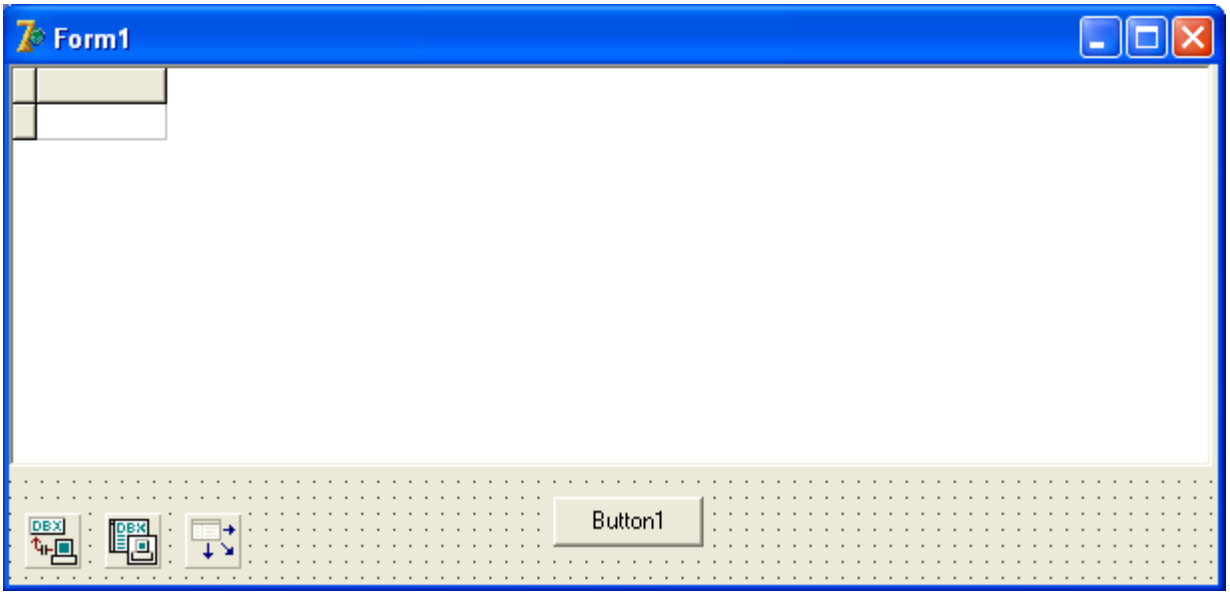

Рис. 24 Внешний вид формы

Сохраняем проект в папку командой File->Save Project As. В эту же папку помещаем dll-библиотеку libmysql.dll (скачать ее можно по ссылке: http://ru.topdll.com/download/libmysql.dll?q=c71850684e8572b1e6d0a4402372495b).

Произведем настройку компонентов.

Выделяем компонент SQLConnection. В контекстном меню выбираем пункт «Edit Connection Properties» (Рис 25):

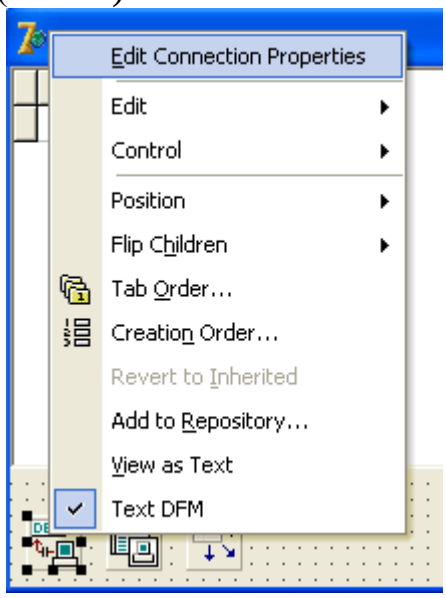

Рис. 25 Контекстное меню

Выбираем в поле «Connection Name» тип MySQLConnection. В настройках подключения (Connection Settings) устанавливаем следующие свойства: HostName – адрес сервера (localhost для локального подключения), Database – имя базы данных, User_Name – имя пользователя при подключении к базе данных, Password – пароль. Нажимаем OK (Рис. 26).

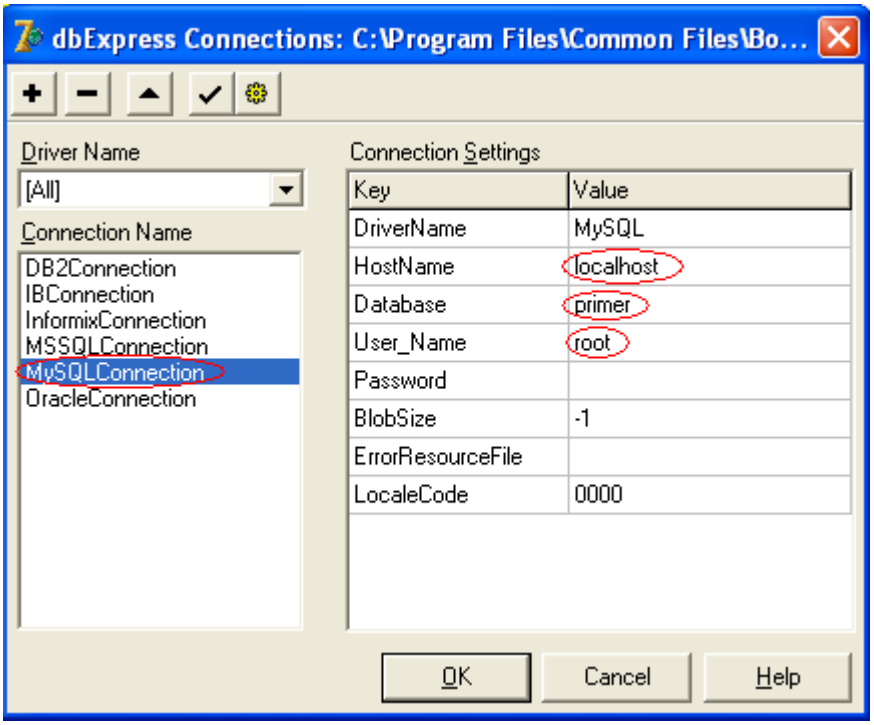

Рис. 26 Настройка соединения с базой данных.

Проверяем, что выделен по-прежнему компонент SQLConnection, устанавливаем в инспекторе объектов (Object Inspector) свойство «LoginPrompt» в false (это позволит отключить запрос пароля при каждом подключении к базе), а также «Connected» в true (Рис. 27).

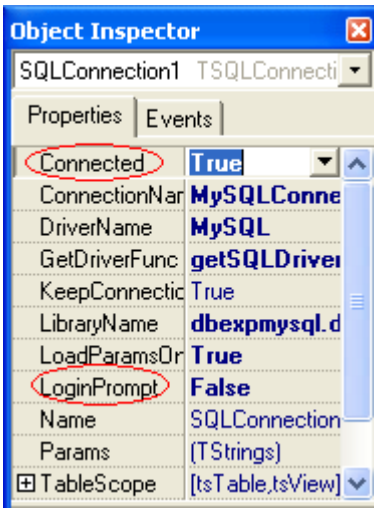

Рис. 27 Окно свойств

Теперь настроим компонент SimpleDataSet. В Object Inspector для свойства «Connection» выберите из выпадающего списка значение SQLConnection1 (рис. 28).

Рис. 28 Свойство Connection

Далее раскрываем свойство «DataSet» и в строке «CommandText» пишем запрос к базе данных. Например, «Select * from zarp» (Рис. 29).

Рис. 29 Свойство CommandText

Запрос можно прописать вручную или с помощью редактора запросов «Command-Text Editor» (для его вызова нужно нажать на ...) (Рис. 30):

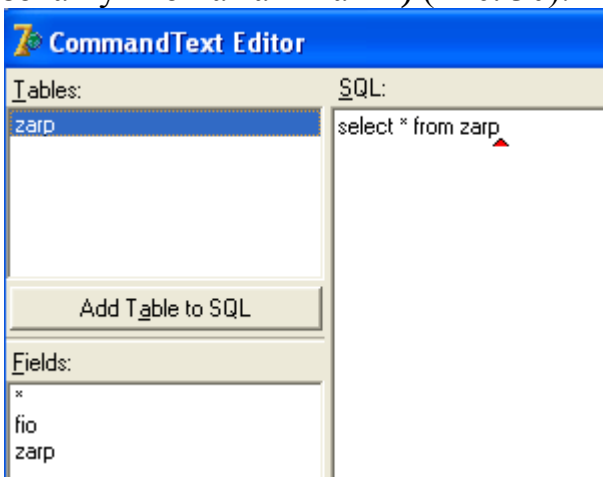

Рис. 30 Создание запросов

Настроим теперь компонент DataSource. Устанавливаем свойство «DataSet» в SimpleDataSet1 (Рис. 31):

Рис. 31 Свойство DataSet

Последний компонент, который нужно настроить – DBGrid. Устанавливаем свойство «DataSource» в DataSource1 (Рис. 32).

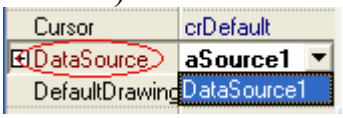

Рис. 32 Свойство DataSource

Программируем кнопку так, чтобы при ее нажатии активировался написанный нами запрос: создаем для кнопки обработчик OnClick прописываем в процедуре код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
DataSource1.DataSet.Active:=true;
end;
```

Запускаем проект на исполнение (F9). Нажимаем на кнопку. Таблица должна заполниться данными из базы. При этом данные, записанные кириллицей, могут отображаться знаками вопроса «?» (Рис. 33).

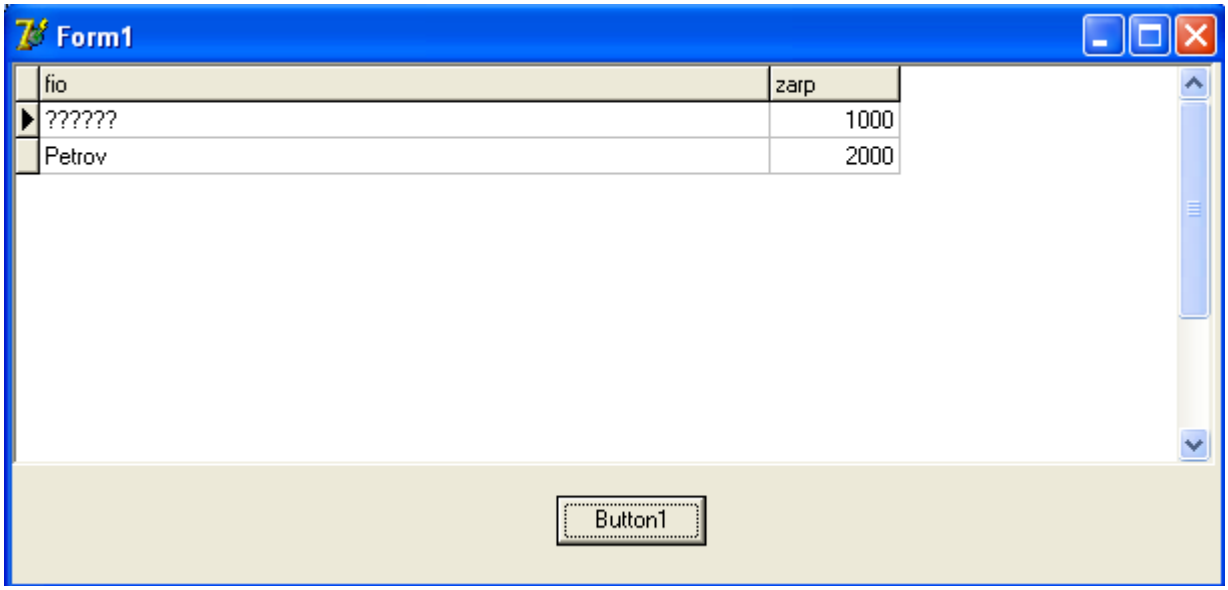

Рис. 33 Вывод таблицы на форму

Для исправления этого необходимо до выполнения запроса установить параметры кодировки. Итак, создаем обработчик события формы OnCreate и прописываем код:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
SQLConnection1.ExecuteDirect(' SET names cp1251');
SQLConnection1.ExecuteDirect(' SET CHARACTER SET cp1251');
end;
```

Запускаем, проверяем работоспособность проекта.

Далее программу можно усложнить, добавив возможность поиска информации, ее сортировки и т.д.

Список литературы:

1. SQL в примерах и задачах ; Учебное пособие/И.Ф Астахова, А.П. Толстобров, В.И. Мельников. – Мн.: Новое знание, 2002.- 176 с.
2. Бабенко Т.А.
   Иллюстрации к лекциям по курс информационные системы: учебно-методическое пособие для студентов / Бабенко Т.А., Бельченко В.Е. Ч.1. Команды отбора и изменения данных.- Армавир: АГПИ, 2005. – 36 с.
3. Бабенко Т.А., Бельченко В.Е.
   Язык SQL в примерах: учебно-методическое пособие. Ч.1. Команды отбора и изменения данных.- Армавир, 2004. – 26 с.
4. Вишневский А., Мамаев Е.
   Microsoft SQL Server 7 для профессионалов.- Санкт-Петербург: Питер, 2001. - 896С.
5. Глушаков С.В. и др.
   Базы данных: Учебный курс. (Домашняя библиотека).- Ростов-на-Дону: Феникс, 2000. – 504 с.
6. Диго С.М.
   Базы данных: проектирование и использование: уч-к.- Москва: Финансы и статистика, 2005. – 595 с.
7. Дунаев В.В.
   Базы данных. Язык SQL для студента: 2-е изд. доп. и перераб. – СПб.: БХВ-Петербург, 2007. – 320 с.
8. Златополский Д.М.
   Сборник заданий на разработку запросов: дидактический материал по теме "Базы данных". - Москва: Чистые пруды, 2005. -32 с.
9. Избачков Ю.С.
   Информационные системы: учебник для вузов / Ю.С. Избачков, В.Н. Петров. - 2-е изд.- Санкт-Петербург: Питер, 2005. – 656 с.
10. Информационные системы и технологии в экономике: учебник. - 2-е изд., доп. и перераб. / под ред. В.И. Лойко.- Москва: Финансы и статистика, 2005. – 416 с.
11. Каба М.
    MYSQL и Perl: коммерческие приложения для Интернета. Учебный курс + Cd.- Санкт - Петербург: Питер, 2001. – 288 с.
12. Калверт Ч.
    Базы данных в Delphi 4. Руководство разработчика.- К: Диасофт, 1999. - 461с.
13. Карпова Т.
    Базы данных. Модели, разработка, реализация. - Санкт-Петербург: Питер, 2002. – 304 с.
14. Максим Кузнецов, Игорь Симдянов
    MySQL 5.- БХВ-Петербург, 2010.- 1024 с.

15. Марков А.С.
Базы данных. Введение в теорию и методологию: уч-к / Марков А.С., Лисовский К.Ю. - Москва: Финансы и статистика, 2004. – 512 с.
16. Плоткин Б.И.
Универсальная алгебра, алгебраическая логика и базы данных. - Москва: Наука, 1991. – 448 с.
17. Редько В.Н., Басараб И.А.
Базы данных и информационные системы. - Москва: Знание, 1987. -31 с.
18. Саукап Р.
Основы Microsoft SQL Server 6.5 / Пер. с англ. - Москва: "Русская редакция"; ТОО "Channel Trading Ltd", 1999. – 704 с.
19. Стив Суэринг, Тим Конверс, Джойс Парк
PHP и MySQL. Библия программиста. – Диалектика, 2010. -912 с.
20. Тоу Д.
Настройка SQL: для профессионалов / Тоу Д.-Санкт-Петербург: Питер, 2004. – 333с.
21. Уткин В.Б., Балдин К.В.
Информационные системы и технологии в экономике: учебник для вузов.- Москва: ЮНИТИ-ДАНА, 2005. – 335 с.
22. Фролов А.В., Фролов Г.В.
Базы данных в интернете: практ. пособие по созданию Web-приложений с базами данных.-Москва: Изд- торг. дом "Рус. редакция", 2000. – 432 с.
23. Шварц Б., Зайцев П., Ткаченко В. и др.
MySQL. Оптимизация производительности.- Символ-Плюс, 2010.- 832 с.