***Лабораторная работа №3***

**Тема:** Знакомство с MySQL. Выборка данных из таблиц.

**Цель:** Получить практические навыки выборки данных из таблиц средствами MySQL.

**Задание:**

1. Изучить теоретическую часть.
2. Грамотно сформулировать и реализовать МИНИМУМ 10 запросов к своей БД (разной степени сложности)
3. Оформить отчет.

***Теоретическая часть***

**Клиентские утилиты**

**Утилита командной строки mysql**

Клиентские утилиты находятся в каталоге **bin** сервера MySQL:

"[C:\Progra](file://C:/Prograra)m Files[\MySQL\MySQL](file:///MySQL/MySQL) Server 5.0\bin"

Сервер **mysqld-nt.exe**, клиентские утилиты **mysql**, **mysqldump**, **mysqlimport** и др.

Утилиты предназначены для работы с сервером и при запуске потребуют указать имя пользователя, под которым мы хотим подключиться к серверу, и пароль. При конфигурировании сервера MySQL мы указали пароль для су­перпользователя **root**. Обычно для получе­ния доступа к серверу MySQL необходимо сообщить клиентской программе и хост, с которым вы хотите соединиться. Например, клиент **mysql** можно запустить следующим образом:

bin>**mysql -h** *host\_name* **-u** *user\_\_name* **–p** *your\_pas*

Здесь ключ **-h** указывает на то, что дальше последует имя хоста, к которому мы хотим подключиться. По умолчанию подключение осуществляется к сер­веру **localhost**. Ключ **-u** означает, что за ним следует имя пользователя, от имени которого осуществляется соединение, а после ключа **-р** указывается пароль. Обратите внимание, что пробела перед паролем нет.

Формат вызова утилиты mysql в нашем случае будет таким:

[C:\Program](file://C:/Program) Files[\MySQL\MySQL](file:///MySQL/MySQL) Server 5.0\bin> **mysql** -u root -p

Утилита mysql может работать в двух режимах — интерактивном и пакетном.

Для работы с утилитой в пакетном режиме следует написать сценарий, со­стоящий из последовательности команд SQL, которые предстоит исполнить для создания базы и таблиц, входящих в базу. Сценарий следует сохранить в тот же каталог **bin** сервера MySQL, из которого мы запускали утилиту mysql. Принято хранить сценарии в файлах с расширением **sql**. Кроме уже рассмотренных команд в скрипте надо предусмотреть возможность того, что в сценарии создания таблицы возможны ошибки. При исправлении ошибок в сценарии вы запускаете его несколько раз, но уже при повторном запуске получаете сообщение о том, что нельзя создать уже существующую таблицу. Значит, надо предварять создание таблицы командой

mysql>**DROP TABLE IF EXISTS** *cars*;

и удалить таблицу, если она уже существует.

Тогда сценарий создания и заполнения таблицы cars будет выглядеть так:

**USE** *taxi*;

**DROP TABLE IF EXISTS** *cars*;

**CREATE TABLE** *cars* (

*model* char(50) NOT NULL,

*madein* year(4) NOT NULL,

*reg\_number* char(12) NOT NULL,

*color* char(15) NOT NULL,

*id* smallint AUTO\_INCREMENT PRIMARY KEY );

**INSERT INTO** *cars* (*model*, *madein*, *reg\_number*, *color*) VALUES ('Volga','2002', 'A788YC78','yellow') , ('Volkswagen','2003','A789YC78','red'), ('Renault','2005','A790YC781,'white');

Скрипт создается в любом текстовом редакторе в кодировке Windows-1251.

При использовании в пакетном режиме сценарии с запросами для создания базы надо копировать в каталог bin и передавать их на вход утилите mysqi при запуске:

[C:\Program](file://C:/Program) Files[\MySQL\MySQL](file:///MySQL/MySQL) Server 5.0\bin>mysql -u root -p < script.sql

Здесь *script.sql* — это сценарий, содержащий команды SQL для создания таб­лиц, записи в них данных и т. п.

**Утилита mysqldump**

Утилита **mysqldump** позволяет получить дамп ("моментальный снимок") со­держимого базы данных или совокупности баз для создания резервной копии или пересылки данных на другой SQL-сервер баз данных. Дамп будет содер­жать набор команд SQL для создания и заполнения таблиц. Откройте окно. Если вы из командной строки дадите такую команду:

[C:\Program](file://C:/Program) Files[\MySQL\MySQL](file:///MySQL/MySQL) Server 5.0\bin > **mysqldump** *имя\_базы*

то распечатаете весь этот набор команд. Чтобы сохранить дамп в файле, дай­те команду перенаправить вывод не на экран, а в файл backup-file.sql:

[C:\Program](file://C:/Program) Files\MySQL\MySQL Server 5.0\bin>**mysqldump** *имя\_базы* > *backup-file.sql*

Если не указывать имена таблиц или использовать параметр **--databases** или --all-databases, то будет получен дамп базы данных в целом (всех баз дан­ных сервера).

Перенеся файл дампа на другой сервер MySQL, можно этот файл передать на вход утилиты MySQL для создания таблиц в базах данных.

Команда

**mysqldump** -u root -р *firma* *cities* > *cities.sql*

приведет к созданию в каталоге bin текстового файла cities.sql следующего содержания.

-- MySQL dump 10.10

-- Host: localhost Database: firma

-- Server version 5.0.15-nt

/\* Здесь пропущены указанные в дампе установки для различных системных параметров \*/

--

-- Table structure for table ‘cities’

--

DROP TABLE IF EXISTS ‘cities’;

CREATE TABLE ‘cities’ (

‘name’ char(30) NOT NULL,

‘description’ text,

‘id’ smallint(6) NOT NULL auto\_increment,

‘country’ char(40) NOT NULL,

PRIMARY KEY (‘id’),

KEY ‘country’ (‘country’),

CONSTRAINT ‘cities\_ibfk\_l’ FOREIGN KEY (‘country’) REFERENCES ‘countries’ (‘name’)

) ENGINE=InnoDB DEFAULT CHARSET=latinl;

--

-- Dumping data for table ‘cities’

--

/\* !40000 ALTER TABLE ‘cities’ DISABLE KEYS \*/;

LOCK TABLES ‘cities’ WRITE;

INSERT INTO ‘cities’ VALUES ('Moscow','City description',1,'Russia');

UNLOCK TABLES;

/\* Здесь пропущены указанные в дампе установки для различных системных параметров \*/

**Утилита mysqlimport**

Утилита **mysqlimport** импортирует текстовые записи в таблицы. На основа­нии имени файла утилита определяет имя таблицы, в которую импортируют­ся данные.

bin> **mysqlimport** параметры *имя\_базы\_данных имя\_\_текстового\_файла*

Для каждого текстового файла, указанного в командной строке, **mysqlimport** удаляет расширение в имени файла и использует его, чтобы определить, в какую таблицу занести содержимое. Например, файлы с именами 'patient.txt', 'patient.text' и 'patient' должны быть все занесены в таблицу с именем ' patient .

Например, для того чтобы импортировать данные из файла new.txt (одноименного с таблицей базы) в базу данных *our\_base*, надо дать следующую команду, а затем ввести пароль:

... bin> mysqlimport -u root -p -local *our\_base* new.txt password:\*\*\*\*\*\*

Каждая строка в файле образует запись в таблице и должна содержать значения полей, разделенные запятыми. Конец записи распознается по символу конца строки текстового файла.

**Запросы к базе данных**

На языке баз данных команды, адресованные базе, называются инструкция­ми либо запросами. Запросом считается такая инструкция, которая возвраща­ет информацию из таблиц в память. Эта информация обычно посылается непосредственно на экран.

Запросы часто отправляются с помощью команды **select**, которая дает инст­рукцию базе данных извлечь информацию из одной или нескольких таблиц.

Вначале необходимо соединиться с сервером MySQL и выбрать базу данных.

mysql>**USE** *taxi*

Запросим данные о модели и цвете автомобилей автопарка:

mysql> **SELECT** *model*, *color* **FROM** *cars*;

Если данных в таблице немного, то можно запросить их все:

mysql>**SELECT** \* **FROM** *table\_name*;

Выясним, какие водители работали за то время, пока ведется учет с помощью базы. Нам нужен только список водителей из таблицы *timetable*, без по­второв.

**distinct** — параметр в запросе **select**, дающий возможность исключить по­вторяющиеся данные из результатов запроса.

mysql> **SELECT DISTINCT** *driver\_number* **FROM** *timetable*;

Результат запроса будет таким:

|  |
| --- |
| driver\_number |
| 1 |
| 2 |

**Запросы с указанием критерия отбора данных**

Таблицы бывают очень большими, а нас могут интересовать только опреде­ленные строки, SQL дает возможность устанавливать критерии для отбора нужных строк. Предложение **where** позволяет определить критерий отбора строк. Операции, допустимые в предложении where, даны в табл. 1.

Узнаем цвет всех автомобилей "Волга":

**SELECT** *color* **FROM** *cars* **WHERE** *model*='Volga';

Таблица 1. Знаки операции сравнения для конструкции **WHERE**

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Название** | **Описание** |
| **=** | Равенство | Проверяет, являются ли два значения равными |
| **>** | Больше | Проверяет, больше ли одно значение дру­гого |
| **<** | Меньше | Проверяет, меньше ли одно значение другого |
| **>=** | Больше или равно | Проверяет, больше или равно одно зна­чение по отношению к другому |
| **<=** | Меньше или равно | Проверяет, меньше или равно одно зна­чение по отношению к другому |
| **! = ИЛИ <>** | Не равно | Проверяет, не равны ли два значения |
| **IS NOT NULL** |  | Проверяет, имеет ли поле значение |
| **IS NULL** |  | Проверяет, не имеет ли поле значение |
| **BETWEEN** | Величина между | Проверяет, значение больше или равно минимальному и меньше или равно мак­симальному |
| **IN** | Значение содержится | Проверяет, содержится ли значение в определенном множестве |
| **NOT IN** | Значение не содер­жится | Проверяет, не содержится ли значение в определенном множестве |
| **LIKE** | Соответствие | Проверяет, отвечает ли значение образ­цу, используя простые механизмы соот­ветствия SQL |
| **NOT LIKE** | Соответствие | Проверяет, не соответствует ли значение образцу |
| **REGEXP** | Регулярное выра­жение | Проверяет, соответствует ли значение регулярному выражению |

Допустим, нам надо узнать, когда выходил на работу водитель с номе­ром 1:

mysql>**SELECT** *use\_on* **FROM** *timetable* **WHERE** *driver\_number* = 1;

Определим номера всех автомобилей "Волга" желтого цвета:

mysql> **SELECT** *reg\_number* **FROM** *cars*

-> **WHERE** *model* = ‘Volga’ **AND** *color* = 'yellow';

Можно проверить несколько критериев сразу, объединяя их операциями **and** или **or**:

mysql> **SELECT** *birth* **FROM** *drivers*

-> **WHERE** *name*='Fedor' **AND** *family\_name*='Sentsov';

Оператор **in** определяет множество значений, которому данное значение мо­жет принадлежать или не принадлежать. Чтобы найти номерные знаки всех автомобилей "Волга" и "Volkswagen", дайте такую команду:

mysql> **SELECT** *reg\_number* **FROM** *cars* **WHERE** *model*

->' **IN** ('Volga', 'Volkswagen');

Оператор **between** задает границы, в которые должно попасть значение, что­бы условие запроса выполнялось. Вы должны ввести ключевое слово **between** с начальным значением, ключевое слово a**n**d и конечное значение. Выясним, кто из водителей устроился на работу в период с 1 января 2001 г. до 1 января 2004 г.:

mysql>**SELECT** *family\_name* **FROM** *drivers*

-> **WHERE** *startdate* **BETWEEN** '2001-01-01' **AND** '2004-01-01';

Оператор **like** использует простой механизм соответствия SQL. Образец может состоять из обычного текста и знака процента (%) для указания совпаде­ния с любым количеством символов. В MySQL соответствия не чувствитель­ны к регистру. Например, шаблон 'Fed%' соответствует любой строке, которая  
начинается с ' Fed ‘.

mysql>**SELECT** \* **FROM** *drivers* **WHERE** *name* **LIKE** 'Fed%';

Ключевое слово **regexp** используется для указания регулярных выражений. MySQL использует регулярные выражения в стиле **POSIX**.

Найдем данные об автомобиле, в номере которого встречается число 788:

mysql>**SELECT** \* **FROM** *cars* **where** *reg\_number*='[A-Z]+788[A-Z0-9]+';

**Группировка данных и агрегатные функции**

Группировка и агрегирование данных совершаются с помощью функций, определенных в MySQL. Описание некоторых функций дано в табл. 2.

Таблица 2. Функции MySQL

|  |  |
| --- | --- |
| **Название** | **Описание** |
| **AVG**(столбец) | Средняя величина значений в определенном столбце |
| **COUNT**(элементы) | При указании столбца выдается число ненулевых значений в этом столбце. Если перед именем столбца указать distinct, то выдается только количество неодинаковых значений |
| **MIN**(столбец) | Минимальное значение в столбце |
| **МАХ**(столбец) | Максимальное значение в столбце |
| **SUM**(столбец) | Сумма значений в столбце |

Агрегатные функции указываются подобно именам полей в запросе **select**, но сами они используют имена поля в качестве аргументов, **sum** и **avg** могут обрабатывать только числовые поля. С функциями **count**, **мах** и **min** могут использоваться и числовые, и символьные поля. Например, чтобы опреде­лить количество водителей в автопарке, можно запросить

mysql>**SELECT** **COUNT**(id) **FROM** *drivers*;

Если надо узнать, сколько водителей работало за отчетное время, то подой­дет команда

mysql>**SELECT** **COUNT** (**DISTINCT** *driver\_number*) **FROM** tim*e*table;

Функция **count**(\*) посчитает число строк в таблице, но если указать **count** (**field**) — посчитать значения в поле field, в котором есть значения null, то пустые поля учтены не будут.

Предложение **group by** позволяет определять подмножество значений в осо­бом поле в терминах другого поля и применять функцию агрегата к подмно­жеству. Это дает вам возможность объединять поля и агрегатные функции в едином предложении **select**.

Посчитаем, сколько автомобилей есть в таксопарке, сгруппировав данные по моделям:

mysql>**SELECT** *model*, **COUNT**(*model*) **FROM** *cars* **GROUP BY** *model*;

Таблицы — это неупорядоченные наборы данных, и выборка из них не обязательно появляется в какой-то определенной последовательности. Для извлечения данных в определенном порядке используется конструкция

**ORDER BY**

Упорядочим сведения о водителях по дате рождения:

mysql>**SELECT** \* **FROM** *drivers* **ORDER BY** *birth*;

Если требуется упорядочить данные по убыванию, следует использовать ключевое слово **desc** (от англ. descending — по убыванию):

mysql>**SELECT** \* **FROM** *drivers* **ORDER BY** *birth* **DESC**;

Конструкция **limit** служит для указания, сколько строк результата следует отображать. Если надо найти двух самых пожилых водителей, то можно применить такую команду:

mysgl>**SELECT** \* **FROM** *drivers* **ORDER BY** *birth* **LIMIT** 2;

Можно использовать два параметра: номер строки, с которой следует начать, и количество строк.

mysql>**SELECT** *family\_name* **FROM** *drivers* **LIMIT** 2,3;

Последний запрос можно интерпретировать так: "Выбрать фамилии водите­лей, в результате отобразить три строки, начиная со строки 2". Нумерация строк начинается с нуля.

**Запросы к двум и более таблицам**

Запрос, который обращен к нескольким записям одной таблицы или к не­скольким таблицам одновременно, называется объединением. В теории баз данных определено несколько типов запросов, чаще других применяется внутреннее объединение (**inner join**).

Рассмотрим запрос, при котором надо выбрать данные из двух таблиц так, чтобы выбирались только те строки, значения которых по соединяемым од­ноименным столбцам совпадают. Например, нам требуется узнать фамилию водителя, который позавчера водил синюю машину. Для того чтобы выбрать все необходимые данные, нам потребуется запросить все три таблицы нашей базы, только тогда мы соберем все нужные сведения.

Нам придется оперировать именами столбцов из разных таблиц. Следует учесть, что полное имя столбца состоит из имени базы данных, таблицы и собственно имени столбца, например, taxi. cars. color.

При работе с одной базой данных из запросов можно исключить упоминание имени этой базы. Если запросы адресуются к одной таблице, то имя таблицы также опускается. Но при объединении необходимо указывать полностью имена таблиц и столбцов.

Часто имена таблиц заменяют короткими псевдонимами (**alias**), указывая их в запросе после ключевого слова **from**. Псевдонимы для таблиц являются стан­дартной частью языка SQL и обычно состоят из одной буквы. Задать псевдо­ним можно с помощью ключевого слова **as**:

mysql> **SELECT** *color* **FROM** *cars* **AS** c;

Псевдонимы удобно использовать в случае, когда имя результирующего столбца слишком длинное или выводятся данные из двух таблиц, в которых есть одноименные столбцы.

Между таблицами существуют связи по определенным столбцам. Столбец *cars.id* связывает таблицу *cars* с таблицей *timetable* через столбец *timetable.car\_number*. Аналогично строится связь между таблицей *drivers* и *timetable*.

Объединение осуществляется по столбцам, образующим связь. При этом вы­бираются одинаковые значения полей, по которым осуществляется связь ме­жду таблицами:

timetable.car\_number=cars.id

и

timetable.driver\_number=drivers.id

Определим дату (данные из столбца *timetable.use\_on*) и фамилию водителя (*drivers. family\_name*), который водил машину определенного цвета

(*cars. color*):

mysql>**SELECT** *t.use\_on*, *d.family\_name*, *c.color*

->**FROM** *timetable* **AS** t, *cars* **AS** c, *drivers* **AS** d

-> WHERE t.car\_number=c.id AND t.driver\_number=d.id;

Выборка данных происходит из таблиц *timetable*, *cars* и *drivers*, для кото­рых определены псевдонимы, сокращающие запись запроса и облегчающие его редактирование. Выбираются только те строки из таблиц *timetable* и *cars*, в которых указан одинаковый учетный номер автомобиля. А из таб­лиц *timetable* и *drivers* выбираются строки, в которых поля в столбце, со­держащем учетный номер водителя, одинаковы.

**Команды обновления и удаления данных в таблицах**

Если необходимо изменить значение поля, то следует использовать команду **update**. В команде надо указать имя используемой таблицы и предложение **set**, определяющее изменение, которое нужно сделать для некоторого столб­ца. Исправим дату выпуска автомобиля "Renault":

mysql>**UPDATE** *cars* **SET** *madein*='2004' **WHERE** *model*='Renault';

Можно изменить значение всех полей столбца, например, при переоценке товаров. Увеличим цены на все товары на 10% с учетом инфляции:

mysql>**UPDATE** *goods* **SET** *price* = price \* 1.1;

Удалить целые строки из таблицы можно с помощью команды **delete**. Уда­лим некоторые строки из таблицы *cars* в связи с продажей старых машин, изготовленных до 2006 г.:

mysql>**DELETE** **FROM** *cars* **where** *madein*<=2006;

Можно удалить все записи из таблицы:

mysql>**DELETE** **FROM** *cars*;

При этом счетчик автомобилей в столбце *id* не сбрасывается, и при вставке новых данных MySQL автоматически присвоит новой записи следующий после последнего удаленного номер.

Для того чтобы очистить таблицу и сбросить счетчик, можно применить команду

mysql>TRUNCATE cars;

Удалить таблицу можно командой **drop table**:

mysql>**DROP TABLE** *timetable*;

При этом надо иметь права на удаление, а также учитывать, что удалить таб­лицу, имеющую родительский ключ, MySQL не позволит, пока не удалена таблица с соответствующим внешним ключом. Удалить можно и всю базу целиком:

mysql>**DROP DATABASE** *taxi*;

**Изменение структуры таблицы**

Оператор **alter table** используется для изменения структуры таблицы. Пре­образования, осуществляемые с помощью этого оператора, представлены в табл. 3.

Таблица 3. Преобразования таблицы с помощью оператора **ALTER TABLE**

|  |  |
| --- | --- |
| **Синтаксис** | **Описание** |
| **ADD** [**COLUMN**] *column\_description* [**FIRST** | **AFTER** *column]* | Добавить новый столбец в указанное место |
| **ADD INDEX** [index] (column, ...) | Добавить индекс в указанный столбец (столбцы) таблицы |
| **ADD PRIMARY KEY** (column, ...) | Сделать указанный столбец (столбцы) первичным ключом таблицы |
| **ALTER [COLUMN**] column {**SET** **DEFAULT** value | **DROP** **DEFAULT**} | Добавить или удалить значение по умолчанию определенного столбца |
| **CHANGE** [**COLUMN**] column new\_column\_description | Изменить столбец с именем column так, чтобы он получил указанное описание. Это можно использовать для изменения имени столбца, поскольку column\_descriptlon включает в себя имя |
| **MODIFY** [**COLUMN**] column\_description | Похоже на change. Используется для изменения типов столбцов, но не имен |
| **DROP** [**COLUMN**] column | Удалить указанный столбец |
| **DROP** **PRIMARY KEY** | Удалить первичный индекс (не столбец!) |
| **DROP INDEX** index | Удалить указанный индекс |
| **RENAME** **table** tbl **to** new\_tbl | Переименовать таблицу |

Допустим, мы решили добавить отсканированные фотографии водителей в базу, чтобы легче было печатать пропуска или другие документы. Ссылки на файлы с фотографиями можно поместить в новый столбец, добавив его в таблицу командой **alter table**:

mysql> **ALTER TABLE** *drivers* **ADD** **COLUMN** *photo* CHAR(50);

В данном примере столбец добавляется в конец таблицы, но можно было бы добавить ключевое слово **first**, тогда столбец стал бы первым, а можно ука­зать **after** *birth*, поместив его тем самым после указанного столбца.

Допустим, вы обнаружили, что поле для указания адреса оказалось мало для вводимых данных: последние символы обрезаются, т. к. адрес длиннее 150 выделенных для него символов. Изменим определение столбца:

mysql> **ALTER** **TABLE** *drivers* **MODIFY** *address* char(200);

**Содержание отчета:**

* Тема;
* Цель;
* Грамотная формулировка КАЖДОГО запроса, скрипт запроса на языке SQL, результат работы запроса;