

Лабораторная работа №2. Типы данных, создание таблиц, редактирование таблиц, создание связей между таблицами.

Цель работы: приобретение практических навыков создания логических и физических моделей данных с помощью MySQL Workbench 5.2 CE.

1. Запустите сервер MySQL, выполняя команду

`mysqld-shareware -standalone` в строке приглашения в каталоге `c:\mysql\bin`.

Более подробно об этом сказано выше, об установке MySQL в Windows.


Или запустите через ярлык в пуске и введите пароль

2. Приглашение изменится на `mysql>`. Введите команду:

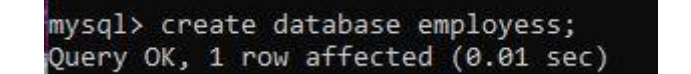
`create database employees;`

(Примечание: команда заканчивается символом точки с запятой).

3. Сервер MySQL должен ответить примерно как на рис. 1



```
Query OK, 1 row affected (0.00 sec)
```



```
mysql> create database employees;  
Query OK, 1 row affected (0.01 sec)  
---
```

Рис. 1. Результат работы команды создания таблицы

[Запрос обработан, изменилась 1 строка (0.01 сек)]

4. Это означает, что была успешно создана база данных. Теперь давайте посмотрим, сколько баз данных имеется в системе. Выполните следующую команду.

`show databases;`

Сервер ответит списком баз данных, как показано на рис. 2

```
+-----+
| Data base |
+-----+
| employees |
| mysql     |
| test      |
+-----+
2 rows in set (0.00 sec)
```

Рис. 2. Просмотр баз данных

Здесь показаны три базы данных, две были созданы MySQL во время установки и вновь созданная база данных **employees**.

5. Для того чтобы выйти из командной строки MySQL введите команду **quit**

Работа с таблицами

Теперь рассмотрим команды MySQL для *создания таблиц* базы данных и выбора базы данных. Базы данных хранят данные в таблицах.

Проще всего таблицы можно представлять себе, как состоящие из строк и столбцов. Каждый столбец определяет данные определенного типа. Строки содержат отдельные записи.

Рассмотрим таблицу, в которой приведены персональные данные некоторых людей:

Таблица. Персональные данные			
Имя	Возраст	Страна	e-mail
Михаил Петров	28	Россия	misha@yandex.ru
Джон Доусон	32	Австралия	j.dow@australia.com
Морис Дрюон	48	Франция	md@france.fr
Снежана	19	Болгария	sneg@bulgaria.com

Приведенная выше таблица содержит четыре столбца, в которых хранятся имя, возраст, страна, и адрес e-mail. Каждая строка содержит данные одного человека. Эта строка называется записью. Чтобы найти страну и адрес e-mail Снежаны, сначала надо выбрать имя в первом столбце, а затем посмотреть содержимое третьего и четвертого столбцов этой же строки.

База данных может содержать множество таблиц, именно таблицы содержат реальные данные.

Следовательно, можно выделить связанные (или несвязанные) данные в различные таблицы. Для базы данных employees определена одна таблица, которая содержит данные компании о сотрудниках, а другая таблица будет содержать персональные данные. Давайте создадим первую таблицу.

Команда SQL для *создания такой таблицы* выглядит следующим образом:

```
CREATE TABLE employee_data
(
  emp_id int unsigned not null auto_increment primary key,
  f_name varchar(20),
  l_name varchar(20),
  title varchar(30),
  age int,
  yos int,
  salary int,
  perks int,
  email varchar(60)
);
```

Примечание: в MySQL команды и имена столбцов не различают регистр символов, однако имена таблиц и баз данных могут зависеть от регистра в связи с используемой платформой (как в Linux). Поэтому можно вместо **CREATE TABLE** использовать **create table**.

За ключевыми словами **CREATE TABLE** следует имя создаваемой таблицы **employee_data**. Каждая строка внутри скобок представляет один столбец. Эти столбцы хранят для каждого сотрудника идентификационный номер (emp_id), фамилию (l_name), имя (f_name), должность (title), возраст (age), стаж работы в компании (yos), зарплату (salary), надбавки (perks), и адрес e-mail (email).

За именем каждого столбца следует тип столбца. Типы столбцов определяют тип данных, которые будет содержать столбец. В данном

примере столбцы **f_name**, **l_name**, **title** и **email** будут содержать текстовые строки, поэтому тип столбца задан как **varchar**, что означает переменное количество символов. Максимальное число символов для столбцов **varchar** определяется числом, заключенным в скобки, которое следует сразу за именем столбца. Столбцы **age**, **yos**, **salary** и **perks** будут содержать числа (целые), поэтому тип столбца задается как **int**. Первый столбец (**emp_id**) содержит идентификационный номер (**id**) сотрудника. Его тип столбца выглядит несколько перегруженным, поэтому рассмотрим его по частям:

- **int**: определяет тип столбца как целое число.
- **unsigned**: определяет, что число будет без знака (положительное целое).
- **not null**: определяет, что значение не может быть **null** (пустым); то есть каждая строка в этом столбце должна иметь значение.
- **auto_increment**: когда MySQL встречается со столбцом с атрибутом **auto_increment**, то генерируется новое значение, которое на единицу больше, чем наибольшее значение в столбце. Поэтому мы не должны задавать для этого столбца значения, MySQL генерирует их самостоятельно. Из этого также следует, что каждое значение в этом столбце будет уникальным.
- **primary key**: помогает при индексировании столбца, что ускоряет поиск значений. Каждое значение должно быть уникально. Ключевой столбец необходим для того, чтобы исключить возможность совпадения данных. Например, два сотрудника могут иметь одно и то же имя, и тогда встанет проблема – как различать этих сотрудников, если не задать им уникальные идентификационные номера. Если имеется столбец с уникальными значениями, то можно легко различить две записи. Лучше всего поручить присваивание уникальных значений самой системе MySQL.

Использование базы данных

База данных **employees** уже создана. Для работы с ней, необходимо её "активировать" или "выбрать". В приглашении **mysql** выполните команду:

SELECT DATABASE();

На экране увидим ответ системы, как показано на рис. 3

```
mysql> SELECT DATABASE();
+-----+
| DATABASE |
+-----+
|          |
+-----+
1 rows in set (0.00 sec)
```

```
mysql> Select database();
+-----+
| database() |
+-----+
| NULL      |
+-----+
1 row in set (0.00 sec)
```

Рис. 3. Выбор базы данных

Это говорит о том, что ни одна база данных не была выбрана. На самом деле всякий раз при работе с клиентом **mysql** необходимо определять, *какая база данных будет использоваться.*

Определить текущую базу данных можно несколькими способами:

- определение имени базы данных при запуске

Введите в приглашении системы следующее:

mysql employees

- определение базы данных с помощью оператора USE в приглашении **mysql**

mysql>USE employees;

```
mysql> USE employess;
Database changed
```

- Определение базы данных с помощью **\u** в приглашении **mysql**
mysql>\u employees;

```
mysql> \stud employees;
-----
C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe Ver 8.0.18 for Win64 on x86_64 (MySQL Community Server - GPL)

Connection id:          21
Current database:
Current user:           root@localhost
SSL:                   Cipher in use is TLS_AES_256_GCM_SHA384
Using delimiter:        ;
Server version:         8.0.18 MySQL Community Server - GPL
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    utf8mb4
Db characterset:        utf8mb4
Client characterset:    cp866
Conn. characterset:     cp866
TCP port:               3306
Uptime:                 21 hours 40 min 45 sec

Threads: 2  Questions: 77  Slow queries: 0  Opens: 212  Flush tables: 3  Open tables: 128  Queries per second avg: 0.000
-----
```

При работе необходимо определять базу данных, которая будет использоваться, иначе MySQL будет порождать ошибку.

Создание таблицы

После выбора базы данных **employees**, выполните в приглашении **mysql** команду **CREATE TABLE**.

```
CREATE TABLE employee_data
(
  emp_id int unsigned not null auto_increment primary key,
  f_name varchar(20),
  l_name varchar(20),
  title varchar(30),
  age int,
  yos int,
  salary int,
  perks int,
  email varchar(60)
);
```

Примечание: нажатие клавиши Enter после ввода первой строки изменяет приглашение **mysql** на **->**. Это означает, что **mysql** понимает, что команда не завершена и приглашает ввести дополнительные операторы. Помните, что каждая команда **mysql** заканчивается точкой с запятой, а каждое объявление столбца отделяется запятой. Можно также при желании ввести всю команду на одной строке.

Вывод на экране должен соответствовать рис. 4.

```
mysql> CREATE TABLE employee data
-> (
-> emp_id int unsigned not null auto_increment primary key,
-> f_name varchar(20),
-> l_name varchar(20),
-> title varchar(30),
-> age int,
-> yos int,
-> salary int,
-> perks int,
-> email varchar(60)
-> );

Query OK, 0 rows affected (0.01 sec)
```

Рис. 4. Создание таблицы

Синтаксис команды CREATE TABLE

Общий формат инструкции **CREATE TABLE** таков:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] имя
[(спецификация, ...)]
[опция, ...]
[[IGNORE | REPLACE] запрос]

Флаг **TEMPORARY** задает *создание временной таблицы*, существующей в течение текущего сеанса. По завершении сеанса таблица удаляется. Временным таблицам можно присваивать имена других таблиц, делая последние временно недоступными.

Спецификатор **IF NOT EXIST** подавляет вывод сообщений об ошибках в случае, если таблица с указанным именем уже существует. Имени таблицы может предшествовать имя базы данных, отделенное точкой. Если это не сделано, таблица будет создана в базе данных, которая установлена по умолчанию.

Чтобы задать имя таблицы с пробелами, необходимо заключить его в обратные кавычки, например **'courses list'**. То же самое нужно будет делать во всех ссылках на таблицу, поскольку пробелы используются для разделения идентификаторов.

Разрешается создавать таблицы без столбцов, однако в большинстве случаев спецификация хотя бы одного столбца все же присутствует. Спецификации столбцов и индексов приводятся в круглых скобках и разделяются запятыми. Формат спецификации следующий:

имя тип

[NOT NULL | NULL]

[DEFAULT значение]

[AUTO_INCREMENT]

[KEY]

[ссылка]

Типы столбцов более подробно будут рассмотрены в разделе "Типы данных столбцов" .

Спецификация типа включает название типа и его размерность. По умолчанию столбцы принимают значения **NULL**. Спецификатор **NOT NULL** запрещает подобное поведение.

У любого столбца есть значение по умолчанию. Если оно не указано, программа MySQL выберет его самостоятельно. Для столбцов, принимающих значения **NULL**, значением по умолчанию будет **NULL**, для строковых столбцов — пустая строка, для численных столбцов — нуль. Изменить эту установку позволяет предложение **DEFAULT**.

Поля-счетчики, создаваемые с помощью флага **AUTO_INCREMENT**, игнорируют значения по умолчанию, так как в них записываются порядковые номера. Тип счетчика должен быть беззнаковым целым. В таблице может присутствовать лишь одно поле-счетчик. Им не обязательно является первичный ключ.

Удаление таблиц

Для того, чтобы удалить таблицу, убедимся сперва, что она существует. Это можно проверить с помощью команды **SHOW TABLES**, как показано на рис. 6.


```
mysql> SHOW TABLES;
+-----+
| Tables in employees |
+-----+
| employee data       |
+-----+
1 rows in set (0.00 sec)
```

Рис. 5. Просмотр таблиц в базе

Для удаления таблицы используется команда **DROP TABLE**, как показано на рис.7.

```
mysql> DROP TABLE employee_data;
Query OK, 0 rows affected (0.01 sec)
```

Рис. 6. Удаление таблицы

Теперь команда **SHOW TABLES**; этой таблицы больше не покажет.

Синтаксис команды DROP TABLE

Инструкция **DROP TABLE** имеет следующий синтаксис:

DROP TABLE [IF EXISTS] таблица [RESTRICT | CASCADE]

Спецификация **IF EXISTS** подавляет вывод сообщения об ошибке, выдаваемого в случае, если заданная таблица не существует. Можно указывать несколько имен таблиц, разделяя их запятыми.

Флаги **RESTRICT** и **CASCADE** предназначены для выполнения сценариев, созданных в других СУБД.

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. Вначале рассмотрены все возможные типы и приведём требования по хранению для каждого типа столбца, затем опишем свойства типов более подробно по каждой категории.

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.

Таблица 4.1. Типы полей MySQL	
<code>TINYINT[(M)] [UNSIGNED] [ZEROFILL]</code>	Очень малое целое число. Диапазон со знаком от -128 до 127. Диапазон без знака от 0 до 255
<code>BIT, BOOL</code>	Синонимы <code>TINYINT(1)</code>
<code>SMALLINT[(M)] [UNSIGNED] [ZEROFILL]</code>	Малое целое число. Диапазон со знаком от -32768 до 32767. Диапазон без знака от 0 до 65535.
<code>MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]</code>	Целое число среднего размера. Диапазон со знаком от -8388608 до 8388607. Диапазон без знака от 0 до 16777215
<code>INT[(M)] [UNSIGNED] [ZEROFILL]</code>	Целое число нормального размера. Диапазон со знаком от -2147483648 до 2147483647. Диапазон без знака от 0 до 4294967295.
<code>INTEGER[(M)] [UNSIGNED] [ZEROFILL]</code>	Синоним для <code>INT</code>
<code>BIGINT[(M)] [UNSIGNED] [ZEROFILL]</code>	Большое целое число. Диапазон со знаком от -9223372036854775808 до 9223372036854775807. Диапазон без знака от 0 до 18446744073709551615
<code>FLOAT(точность) [UNSIGNED] [ZEROFILL]</code>	Число с плавающей точкой. Атрибут точности может иметь значение ≤ 24 для числа с плавающей точкой обычной (одинарной) точности и между 25 и 53 - для числа с плавающей точкой удвоенной точности. Эти типы данных сходны с типами <code>FLOAT</code> и <code>DOUBLE</code> , описанными

	<p>ниже. FLOAT (X) относится к тому же интервалу, что и соответствующие типы FLOAT и DOUBLE, но диапазон значений и количество десятичных знаков не определены.</p>
FLOAT [(M,D)] [UNSIGNED] [ZEROFILL]	<p>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 $X \leq 24$, справедливы для числа с плавающей точкой обычной точности.</p>
DOUBLE [(M,D)] [UNSIGNED] [ZEROFILL]	<p>Число с плавающей точкой удвоенной точности нормального размера. Допустимые значения: от -1,7976931348623157E+308 до -2,2250738585072014E-308, 0, и от 2,2250738585072014E-308 до 1,7976931348623157E+308. Если указан атрибут UNSIGNED, отрицательные значения недопустимы. Атрибут M указывает количество выводимых пользователю знаков, а атрибут D - количество разрядов, следующих за десятичной точкой. Обозначение DOUBLE без указания аргументов или запись вида DOUBLE (X), где $25 \leq X \leq 53$, справедливы для числа с плавающей точкой двойной точности.</p>
DECIMAL [(M[,D])] [UNSIGNED] [ZEROFILL] или DEC [(M[,D])] [UNSIGNED] [ZEROFILL] или NUMERIC [(M[,D])] [UNSIGNED] [ZEROFILL]	<p>"Неупакованное" число с плавающей точкой. Ведет себя подобно столбцу CHAR, содержащему цифровое значение. Термин "неупакованное" означает, что число хранится в виде строки и при этом для каждого десятичного знака используется один символ. Разделительный знак десятичных разрядов, а также знак '-' для отрицательных чисел не учитываются в M (но место для них зарезервировано). Если атрибут D равен 0, величины будут представлены без десятичного знака, т.е. без дробной части. Максимальный интервал значений типа DECIMAL тот же, что и для типа DOUBLE, но действительный интервал для конкретного столбца DECIMAL может быть ограничен выбором значений</p>

	атрибутов M и D . Если указан атрибут UNSIGNED , отрицательные значения недопустимы. Если атрибут D не указан, его значение по умолчанию равно 0. Если не указан M , его значение по умолчанию равно 10.
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.
DATETIME	Комбинация даты и времени. Поддерживается интервал от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'. MySQL выводит значения DATETIME в формате 'YYYY-MM-DD HH:MM:SS', но можно устанавливать значения в столбце DATETIME , используя как строки, так и числа.
TIMESTAMP [(M)]	Временная метка. Интервал от '1970-01-01 00:00:00' до некоторого значения времени в 2037 году. MySQL выводит значения TIMESTAMP в форматах YYYYMMDDHHMMSS , YYMMDDHHMMSS , YYYYMMDD или YYMMDD в зависимости от значений M : 14 (или отсутствующее), 12, 8, или 6; но можно также устанавливать значения в столбце TIMESTAMP , используя как строки, так и числа. Столбец TIMESTAMP полезен для записи даты и времени при выполнении операций INSERT или UPDATE , так как при этом автоматически вносятся значения даты и времени самой последней операции, если эти величины не введены программой. Можно также устанавливать текущее значение даты и времени, задавая значение NULL .
TIME	Время. Интервал от '-838:59:59' до '838:59:59'. MySQL выводит значения TIME в формате 'HH:MM:SS', но можно устанавливать значения в столбце TIME , используя как строки, так и числа.
YEAR [(2 4)]	Год в двухзначном или четырехзначном форматах (по умолчанию формат четырехзначный). Допустимы следующие значения: с 1901 по 2155, 0000 для четырехзначного формата года и 1970-2069 при использовании двухзначного формата (70-69). MySQL выводит значения YEAR в формате YYYY , но можно задавать значения в столбце YEAR , используя как строки, так и

	числа.
<code>[NATIONAL] CHAR(M) [BINARY]</code>	Строка фиксированной длины, при хранении всегда дополняется пробелами в конце строки до заданного размера. Диапазон аргумента <code>M</code> составляет от 0 до 255 символов. Концевые пробелы удаляются при выводе значения. Если не задан атрибут чувствительности к регистру <code>BINARY</code> , то величины <code>CHAR</code> сортируются и сравниваются как независимые от регистра в соответствии с установленным по умолчанию алфавитом. Атрибут <code>NATIONAL CHAR</code> (или его эквивалентная краткая форма <code>NCHAR</code>) представляет собой принятый в ANSI SQL способ указания, что в столбце <code>CHAR</code> должен использоваться установленный по умолчанию набор символов (<code>CHARACTER</code>).
<code>CHAR</code>	Это синоним для <code>CHAR(1)</code> .
<code>[NATIONAL] VARCHAR(M) [BINARY]</code>	Строка переменной длины. Примечание: концевые пробелы удаляются при сохранении значения (в этом заключается отличие от спецификации ANSI SQL). Диапазон аргумента <code>M</code> составляет от 0 до 255 символов. Если не задан атрибут чувствительности к регистру <code>BINARY</code> , то величины <code>VARCHAR</code> сортируются и сравниваются как независимые от регистра.
<code>TINYBLOB, TINYTEXT</code>	Столбец типа <code>BLOB</code> или <code>TEXT</code> с максимальной длиной 255 символов.
<code>BLOB, TEXT</code>	Столбец типа <code>BLOB</code> или <code>TEXT</code> с максимальной длиной 65535 символов.
<code>MEDIUMBLOB, MEDIUMTEXT</code>	Столбец типа <code>BLOB</code> или <code>TEXT</code> с максимальной длиной 16777215 символов.
<code>LONGBLOB, LONGTEXT</code>	Столбец типа <code>BLOB</code> или <code>TEXT</code> с максимальной длиной 4294967295 символов.
<code>ENUM('значение1','значение2',...)</code>	Перечисляемый тип данных. Объект строки может иметь только одно значение, выбранное из заданного списка величин <code>'значение1', 'значение2', ..., NULL</code> или специальная величина ошибки <code>""</code> . Список <code>ENUM</code> может содержать максимум 65535 различных величин
<code>SET('значение1','значение2',...)</code>	Набор. Объект строки может иметь ноль или более значений, каждое из которых должно быть выбрано из заданного списка величин <code>'значение1', 'значение2', ...</code> . Список <code>SET</code> может содержать максимум 64 элемента.

Команда CREATE DATABASE

CREATE DATABASE создает базу данных с указанным именем. Чтобы использовать эту команду, необходимо иметь CREATE привилегии для базы данных. CREATE SCHEMA является синонимом CREATE DATABASE в MySQL 5.0.2.

Пример

Создание базы данных с именем 'test_database'.

CREATE DATABASE test_database

или

CREATE SCHEMA test_database

Команда USE

USE сообщает MySQL, что указанная база данных является выбранной (текущей). Таким образом, команды применяются именно к этой базе.

Пример

Указать, что база данных с именем 'test_database' является текущей.

USE test_database;

```
mysql> use test_database
Database changed
```

Команда SHOW DATABASES

SHOW DATABASES показывает список баз данных на сервере MySQL. SHOW SCHEMAS является синонимом SHOW DATABASES в MySQL 5.0.2.

Команда SHOW CREATE DATABASE

Показывает команду CREATE DATABASE, с помощью которой была создана база данных.

Пример

Вывод информации о команде, которой была создана база данных 'test_database'.

SHOW CREATE DATABASE test_database;

```
mysql> show create database test_database;
+-----+
| Database | Create Database |
+-----+
| test_database | CREATE DATABASE `test_database` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */ |
+-----+
1 row in set (0.03 sec)

mysql>
```

Команда SHOW TABLES

Показывает таблицы определенной базы данных.

Пример

Вывод таблиц базы данных 'test_database'.

SHOW TABLES FROM test_database

Команда CREATE TABLE

CREATE TABLE создает таблицу с заданным именем. Чтобы использовать эту команду, необходимо иметь CREATE привилегии.

Пример

Создание таблицы 'test_table' со столбцами id (целочисленного типа) и data (строкового типа, не более 100 символов).

CREATE TABLE test_table

(
id INT,
data VARCHAR(100)
);

Команда ALTER TABLE

ALTER TABLE изменяет структуру таблицы. Например, вы можете добавлять или удалять столбцы, изменять тип существующих столбцов или переименовывать столбцы, либо саму таблицу. Вы также можете изменить такие характеристики, как механизм хранения, используемый для таблицы и др.

Пример

Переименование таблицы 'test_table' в 't_table'.

ALTER TABLE test_table RENAME t_table;

Изменение типа столбца id в таблице test_table на TINYINT и установление свойства NOT NULL, переименование столбца строкового типа data в newData и изменение размера на 200.

```
ALTER TABLE test_table MODIFY id TINYINT NOT NULL, CHANGE data  
newData VARCHAR(200);
```

Добавление столбца date типа TIMESTAMP в таблицу test_table.

```
ALTER TABLE test_table ADD date TIMESTAMP;
```

Удаление столбца date из таблицы test_table.

```
ALTER TABLE test_table DROP COLUMN date;
```

Добавление столбца newID типа INT UNSIGNED в таблицу test_table.

При этом происходит индексирование столбца как PRIMARY KEY, чтобы было возможным использовать свойство AUTO_INCREMENT. Свойство NOT NULL устанавливается, чтобы использовать столбец в качестве первичного ключа.

```
ALTER TABLE test_table ADD newID INT UNSIGNED NOT NULL  
AUTO_INCREMENT, ADD PRIMARY KEY (newID)
```

Команда DROP TABLE

DROP TABLE удаляет одну или несколько таблиц. Чтобы использовать эту команду, необходимо иметь DROP привилегии.

Пример

Удаление таблиц B, C, A из базы данных, если они существуют.

```
DROP TABLE IF EXISTS B, C, A;
```

Удаление таблицы 'test_table' из текущей базы данных.

```
DROP TABLE test_table;
```

Команда INSERT

INSERT добавляет новую строку в существующую таблицу.

Пример

Добавление строки, ('1', 'Иван', 'Иванов') в соответствующие столбцы таблицы 'new_table' базы данных 'new_schema'.


```
INSERT INTO `new_schema`.`new_table` (`id`, `name`, `surname`) VALUES ('1',  
'Иван', 'Иванов');
```

Команда UPDATE

UPDATE обновляет строку в существующей таблице.

Пример

Обновление данных в строке с id=1 в соответствующих столбцах таблицы 'new_table' базы данных 'new_schema'.

```
UPDATE `new_schema`.`new_table` SET `name`='Владимир',  
`surname`='Владимиров' WHERE `id`='1';
```

--

```
mysql> UPDATE test.test_table set name='Иван', surname='Иванов', age=45 WHERE id=1;
```

--

Команда DELETE

DELETE удаляет данные из существующей таблицы.

Пример

Удаление данных из таблицы 'new_table' базы данных 'new_schema', где id=1.

```
DELETE FROM `new_schema`.`new_table` WHERE `id`='1';
```

Самостоятельное задание

1. Создайте таблицу, содержащую два столбца с именами и фамилиями сотрудников.

```
mysql> SELECT f_name, l_name from employee_data;
```

f_name	l_name
Михаил	Петров
Иван	Гусев
Григорий	Распутин
Мария	Путина
Елена	Арго
Федер	Круглов
Иван	Макаров
Эдуард	Сахаров
Антон	Павлов
Кирилл	Раков
Павел	Силин
Артур	Харон
Римма	Чашина
Рита	Белова
Денис	Горшков
Ольга	Лисина
Елена	Кароян
Василий	Курица
Вера	Козлова
Степан	Аидов
Петр	Ключник

21 rows in set (0.00 sec)

Рис. 7. Вывод данных из таблицы

2. Добавьте столбцы с информацией: возраст, должность, стаж работы, зарплата.

3. Замените у нескольких сотрудников данные: должность, зарплата.