MySQL

Урок 2

SQL - команды DDL

Научимся, пользуясь SQL, создавать, изменять и удалять БД и объекты БД.

Data Definition Language (DDL)

Data Definition Language (DDL) – это язык, с помощью которого можно описать структуру данных в БД. Пользуясь DDL, можно создать и удалить БД, добавить таблицу с любой структурой и связями, изменить и удалить таблицу.

Команды делятся на три вида: create, alter, drop. С помощью этих команд можно провести любые операции с БД и её составляющими.

Далее мы обзорно рассмотрим синтаксис всех трех команд.

Создание базы данных

Для создания БД используется следующая команда:

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

Команда создает базу данных с именем «db_name».

Ключ [IF NOT EXISTS] используется для проверки на существование базы данных. Если ключ не указан и база данных существует, это приведет к возникновению ошибки.

Создание таблицы

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    [partition_options]

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options]
    [partition_options]
    [IGNORE | REPLACE]
    [AS] query_expression
```

Команда позволяет создать таблицу с именем «tbl_name». Также как и в случае создания БД, есть флаг проверки [IF NOT EXISTS], который позволяет проверить на существование в БД таблицы с аналогичным именем.

[TEMPORARY] позволяет создать таблицу только на текущую сессию. Сразу после завершения сессии таблица будет удалена.

Объявление полей в БД

```
column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
      [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
      [COMMENT 'string']
      [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
      [STORAGE {DISK|MEMORY|DEFAULT}]
      [reference_definition]
  | data_type [GENERATED ALWAYS] AS (expression)
      [VIRTUAL | STORED] [UNIQUE [KEY]] [COMMENT comment]
      [NOT NULL | NULL] [[PRIMARY] KEY]
data_type:
    BIT[(length)]
  | TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  | SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
  | INT[(length)] [UNSIGNED] [ZEROFILL]
  | INTEGER[(length)] [UNSIGNED] [ZEROFILL]
  | BIGINT[(length)] [UNSIGNED] [ZEROFILL]
  | REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DECIMAL[(length[,decimals])] [UNSIGNED] [ZEROFILL]
  | NUMERIC[(length[,decimals])] [UNSIGNED] [ZEROFILL]
  | DATE
  | TIME[(fsp)]
  | TIMESTAMP[(fsp)]
  | DATETIME[(fsp)]
  | YEAR
  | CHAR[(length)] [BINARY]
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | VARCHAR(length) [BINARY]
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | BINARY[(length)]
  | VARBINARY(length)
  | TINYBLOB
  | BLOB
  | MEDIUMBLOB
  | LONGBLOB
  | TINYTEXT [BINARY]
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | TEXT [BINARY]
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | MEDIUMTEXT [BINARY]
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | LONGTEXT [BINARY]
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | ENUM(value1,value2,value3,...)
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | SET(value1,value2,value3,...)
      [CHARACTER SET charset_name] [COLLATE collation_name]
  | JSON
  | spatial_type
```

При объявлении столбца объявляется тип данных, а также ряд дополнительных опций data_type [NOT NULL | NULL] [DEFAULT default_value].

[NOT NULL | NULL] используется для проверки в колонке нулевого/ненулевого значения. При добавлении данных поле может быть обязательным, для этого мы должны использовать флаг [NOT NULL].

[DEFAULT default_value] используется для задания полю значения по умолчанию.

При объявлении полей в БД мы можем сразу присвоить значение ключевым полям – [UNIQUE [KEY] | [PRIMARY] KEY, также поле можно создать с опцией [AUTO_INCREMENT], что позволит каждый раз при добавлении новых данных увеличивать значение поля на единицу. Используется для отслеживания уникальности первичного ключа.

[COMMENT 'string'] позволяет написать комментарий к полю в БД. Очень удобно просматривать комментарии при работе с БД. Комментировать необходимо действительно важные значения.

[STORAGE {DISK|MEMORY|DEFAULT}] определяет носитель, куда будут записываться данные, – на диск или в оперативную память.

Выбор типа данных для колонки

Типы данных мы рассматривали в предыдущем уроке, сегодня надо обратить внимание только на синтаксис DDL.

Объявление ключей и индексов

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    { LIKE old_tbl_name | (LIKE old_tbl_name) }
create_definition:
    col_name column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
      [index_option] ...
  | {INDEX|KEY} [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY]
      [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] FOREIGN KEY
      [index_name] (index_col_name,...) reference_definition
  | CHECK (expr)
index_col_name:
    col_name [(length)] [ASC | DESC]
index_type:
    USING {BTREE | HASH}
reference_definition:
    REFERENCES tbl_name (index_col_name,...)
      [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
      [ON DELETE reference_option]
      [ON UPDATE reference_option]
reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION
```

При объявлении ключа мы должны задать его имя [CONSTRAINT [symbol]], далее объявляется тип ключа – PRIMARY KEY | UNIQUE [INDEX|KEY] | FOREIGN KEY.

Так же можно объявить индекс.

```
{INDEX|KEY} [index_name] [index_type] (index_col_name,...)
index_type:
    USING {BTREE | HASH}
```

Индексы используются для быстрого поиска, мы это разбирали в предыдущем уроке.

Внешний ключ

```
[CONSTRAINT [symbol]] FOREIGN KEY
      [index_name] (index_col_name,...) reference_definition
reference_definition:
    REFERENCES tbl_name (index_col_name,...)
      [ON DELETE reference_option]
      [ON UPDATE reference_option]
reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION
```

Чтобы внешний ключ был создан корректно, необходимо, чтобы тип данных колонки в таблице совпадал с типом данных колонки, на которую этот ключ указывает.

Объявление самого внешнего ключа не отличается от объявления других ключей, только в дополнение к ним имеется ряд опций.

REFERENCES tbl_name (index_col_name, ...) – указывает, на какую таблицу ссылается внешний ключ

ON DELETE, ON UPDATE – операции, которые выполняются при удалении или изменении связанной записи – для этого существуют опции RESTRICT | CASCADE | SET NULL | NO ACTION – выбросить ошибку, удалить связанные данные, установить данные в значение NULL, не выполнять никаких действий.

Благодаря внешнему ключу контролируется целостность связанных данных.

Alter table

Данная команда позволяет изменить структуру таблицы – добавить, изменить и удалить колонку в таблице.

```
ALTER [IGNORE] TABLE tbl_name
    [alter_specification [, alter_specification] ...]
    [partition_options]
alter_specification:
    table_options
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name ]
  | ADD [COLUMN] (col_name column_definition,...)
  | ADD {INDEX|KEY} [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [INDEX|KEY] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD SPATIAL [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [index_name] (index_col_name,...)
        reference_definition
  | ALGORITHM [=] {DEFAULT|INPLACE|COPY}
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST|AFTER col_name]
  | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
  | MODIFY [COLUMN] col_name column_definitiondrop
        [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name - удаление
  | DROP PRIMARY KEY
  | DROP {INDEX|KEY} index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS - включает и выключает внешние ключи
  | ENABLE KEYSrename
  | RENAME [TO|AS] new_tbl_name - переименовать
  | RENAME {INDEX|KEY} old_index_name TO new_index_name
  | ORDER BY col_name [, col_name] … - сортировка по названию
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | FORCE
  | {WITHOUT|WITH} VALIDATION
  | ADD PARTITION (partition_definition)
  | DROP PARTITION partition_names
  | DISCARD PARTITION {partition_names | ALL} TABLESPACE
  | IMPORT PARTITION {partition_names | ALL} TABLESPACE
  | TRUNCATE PARTITION {partition_names | ALL}
  | COALESCE PARTITION number
  | REORGANIZE PARTITION partition_names INTO (partition_definitions)
   | EXCHANGE PARTITION partition_name WITH TABLE tbl_name [{WITH|WITHOUT}
VALIDATION]
  | ANALYZE PARTITION {partition_names | ALL}
  | CHECK PARTITION {partition_names | ALL}
  | OPTIMIZE PARTITION {partition_names | ALL}
  | REBUILD PARTITION {partition_names | ALL}
```

```
  | REPAIR PARTITION {partition_names | ALL}
  | REMOVE PARTITIONING
  | UPGRADE PARTITIONING
index_col_name:
    col_name [(length)] [ASC | DESC]
index_type:
    USING {BTREE | HASH}
index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
table_options:
    table_option [[,] table_option] ...  (see CREATE TABLE options)
```

Как видно из спецификации команды, мы можем осуществить любые операции по изменению таблицы, пользуясь данной командой.

Команда может выступать в нескольких ипостасях в зависимости от опций:

- ADD – добавить в таблицу колонку, ключ, индекс.
- CHANGE – изменить имя колонки.
- DROP – удалить колонку, ключ, индекс.
- RENAME – сменить имя таблице.
- CONVERT TO CHARACTER SET – сменить кодировку.

Остальные опции используются для манипуляций механизмами, которые мы рассмотрим в других уроках.

Drop

Синтаксис для удаления БД.

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

Синтаксис для удаления таблицы.

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

Оператор DROP используется для удаления БД или таблицы. Синтаксис очень простой.

[IF EXISTS] перед удалением проверяет наличие данной БД или таблицы.

[RESTRICT | CASCADE] удалит связанные таблицы или выдаст ошибку.

Практическая работа

Создадим БД «Страны и города мира» с помощью готовых скриптов, приложенных к уроку. Научимся делать импорт SQL-файлов.

- Подключаемся к MySQL с помощью команды:

```
mysql -u[username] -p[password]
```

- Можно использовать пользователя root, которого мы создали при установке MySQL:

```
mysql -uroot -p[password]
```

- Для того, чтобы создать базу данных используется команда:

```
CREATE DATABASE geodata;
```

- Выходим из MySQL shell:

```
exit;
```

- Рассмотрим подробнее файлы для создания БД:
  - Tables.sql – файл, который содержит в себе структуру БД, команды DDL для создания схемы.
  - _cities.sql, _countries.sql, _regions.sql – файлы необходимые для импорта данных.

  На этом уроке мы импортируем схему и данные. Для этого используется команда:

```
mysql -u[username] -p[password] [database name] < file.sql
```

Практическое задание

1. Имеющаяся у нас схема не очень годится для работы. Нужно привести её в нормальный вид, таблицы должны выглядеть следующим образом:

Таблица _countries

FIELD	TYPE	NULL	KEY	EXTRA
id	INT	NOT	Primary Key	auto_increment
title	VARCHAR(150)	NOT	INDEX	

Таблица _regions

FIELD	TYPE	NULL	KEY	EXTRA
id	INT	NO	Primary Key	auto_increment
country_id	INT	NO	Foreign Key _countries (id)	
title	VARCHAR(150)	NO	INDEX	

Таблица _cities

FIELD	TYPE	NULL	KEY	EXTRA
id	INT	NO	Primary Key	auto_increment
country_id	INT	NO	Foreign Key _countries (id)	
important	TINYINT(1)	NO		
region_id	INT	NO	Foreign Key _regions (id)	
title	VARCHAR(150)	NO	INDEX	

Самый простой способ – написать команды ALTER TABLE в отдельном файле с расширением .sql и импортировать его:

```
mysql -u[username] -p[password] [database name] < file.sql
```

На изменение схемы может уйти очень много времени (до 1 часа), так что не пугайтесь, если команда заставит вас долго ждать.

Для проверки результатов работы могут быть полезны следующие команды:

Показать команду DDL, с помощью которой таблица была создана:

```
SHOW CREATE TABLE `<yourtable>`;
```

Показать структуру таблицы:

```
SHOW COLUMNS FROM `<yourtable>`;
```

Дополнительные материалы

1. http://www.mysqltutorial.org/

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. http://dev.mysql.com/doc/refman/5.7/en/alter-table.html
2. http://dev.mysql.com/doc/refman/5.7/en/create-table.html