链滴

3-SQL 语言 -DDL，DML

作者：Carey

原文链接：https://ld246.com/article/1614770369223

来源网站：链滴

3 SQL 语言

3.1 关系型数据库的常见组件

- 数据库：database
- 表：table，行：row 列：column
- 索引：index
- 视图：view
- 存储过程：procedure
- 存储函数：function
- 触发器：trigger
- 事件调度器：event scheduler，任务计划
- 用户：user
- 权限：privilege

3.2 SQL语言的兴起与语法标准

目前，所有主要的关系数据库管理系统支持某些形式的SQL，大部分数据库至少遵守ANSI SQL89标，虽然有这一标准的存在，但大部分的SQL代码在不同的数据库系统中并不具有完全的跨平台性业内准

微软和Sybase的T-SQL，Oracle的PL/SQL

3.2.1 SQL 语言规范

在数据库系统中，SQL 语句不区分大小写，建议用大写

SQL语句可单行或多行书写，默认以 " ; " 结尾

关键词不能跨多行或简写

用空格和TAB 缩进来提高语句的可读性

子句通常位于独立行，便于编辑，提高可读性

注释：

- SQL标准：

```
#单行注释，注意有空格
-- 注释内容

#多行注释
/*注释内容
注释内容
注释内容*/
```

- MySQL注释：

```
# 注释内容
```

3.2.2 数据库对象和命名

数据库的组件(对象)：

- 数据库、表、索引、视图、用户、存储过程、函数、触发器、事件调度器等

命名规则：

- 必须以字母开头，后续可以包括字母,数字和三个特殊字符（# _ $）
- 不要使用MySQL的保留字

3.2.3 SQL语句分类

- DDL: Data Defination Language 数据定义语言
  - CREATE，DROP，ALTER
- DML: Data Manipulation Language 数据操纵语言
  - INSERT，DELETE，UPDATE
- DQL：Data Query Language 数据查询语言
  - SELECT
- DCL：Data Control Language 数据控制语言
  - GRANT，REVOKE

● 软件开发：CRUD

3.2.4 SQL语句构成

关健字Keyword组成子句clause，多条clause组成语句

示例：

```
SELECT *                #SELECT子句
FROM products           #FROM子句
WHERE price>666         #WHERE子句
```

说明：一组SQL语句由三个子句构成，SELECT,FROM和WHERE是关键字

获取SQL 命令使用帮助：

官方帮助：https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html

```
mysql> HELP KEYWORD
```

3.2.5 字符集和排序

早期MySQL版本默认为latin1，从MySQL8.0开始默认字符集已经为 utf8mb4

查看支持所有字符集：

```
11:37:27(root@localhost) [(none)]> show character set;
#特别注意mysql中utf8并不是真正的uft8
正确的utf8最大应该占4个字节而mysql中utf8占3个字节，所以会导致一些生僻字显示不了
如要使用utf8应使用utf8mb4
utf8     | UTF-8 Unicode                 | utf8_general_ci     |      3
```

查看支持所有排序规则：

```
11:37:42(root@localhost) [(none)]> show collation;
```

查看当前使用的排序规则

```
11:38:29(root@localhost) [(none)]> show variables like 'collation%';
```

设置服务器默认的字符集

```
[11:44:27 root@centos8 ~]#vim /etc/my.cnf
[mysqld]
character-set-server=utf8
```

设置客户端默认的字符集

```
#针对mysql客户端
[mysql]
default-character-set=utf8
#针对所有mysql客户端
[client]
default-character-set=utf8
```

范例：字符集和相关文件

```
11:47:48(root@localhost) [(none)]> show character set;

[11:48:20 root@centos8 ~]#ll /usr/share/mysql/charsets/
```

查看当前字符集的使用情况

```
11:49:01(root@localhost) [(none)]> show variables like 'character%';
```

3.3 管理数据库

3.3.1 创建数据库

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] 'DB_NAME'
CHARACTER SET 'character set name'
COLLATE 'collate name';
```

范例：

```
#创建数据库
11:49:20(root@localhost) [(none)]> create database db1;
Query OK, 1 row affected (0.00 sec)
#查看你创建的数据库
11:51:58(root@localhost) [(none)]> 11:51:58(root@localhost) [(none)]> show create database
db1;
+----------+-----------------------------------------------------------------------------------
------+
| Database | Create Database                                                                   |
+----------+-----------------------------------------------------------------------------------
------+
| db1      | CREATE DATABASE `db1` /*!40100 DEFAULT CHARACTER SET utf8 */ /*!80016 DEF
ULT ENCRYPTION='N' */ |
+----------+-----------------------------------------------------------------------------------
------+
1 row in set (0.00 sec)
#使用的字符集和排序规则存放位置
[root@centos8 ~]#cat /var/lib/mysql/db1/db.opt
default-character-set=latin1
default-collation=latin1_swedish_ci

11:55:31(root@localhost) [(none)]> create database db1;
ERROR 1007 (HY000): Can't create database 'db1'; database exists
11:55:41(root@localhost) [(none)]> create database IF NOT EXISTS db1;
Query OK, 1 row affected, 1 warning (0.00 sec)

11:56:22(root@localhost) [(none)]> show warnings;
+-------+------+-------------------------------------------+
| Level | Code | Message                                   |
+-------+------+-------------------------------------------+
| Note  | 1007 | Can't create database 'db1'; database exists |
+-------+------+-------------------------------------------+
1 row in set (0.00 sec)
```

范例：指定字符集创建新数据库

```
11:56:33(root@localhost) [(none)]> create database IF NOT EXISTS db2 CHARACTER SET 'utf
mb4';
Query OK, 1 row affected (0.00 sec)
11:58:17(root@localhost) [(none)]> 11:58:17(root@localhost) [(none)]> show create database
db2;
+----------+-----------------------------------------------------------------------------------------
-----------------------------------+
| Database | Create Database
   |
+----------+-----------------------------------------------------------------------------------------
-----------------------------------+
| db2      | CREATE DATABASE `db2` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf
mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */ |
+----------+-----------------------------------------------------------------------------------------
-----------------------------------+
1 row in set (0.00 sec)
```

范例: 创建数据库指定字符集，并且指定排序规则

```
12:00:16(root@localhost) [(none)]> create database zabbix character set utf8 collate utf8_bin;
```

3.3.2 修改数据库

```
ALTER DATABASE DB_NAME character set utf8;
```

范例：

```
12:04:43(root@localhost) [(none)]> alter database db1 character set utf8mb4;
Query OK, 1 row affected (0.00 sec)

12:05:14(root@localhost) [(none)]> show create database db1;
+----------+-----------------------------------------------------------------------------------------
-----------------------------------+
| Database | Create Database
   |
+----------+-----------------------------------------------------------------------------------------
-----------------------------------+
| db1      | CREATE DATABASE `db1` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf
mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */ |
+----------+-----------------------------------------------------------------------------------------
-----------------------------------+
1 row in set (0.00 sec)
```

3.3.3 删除数据库

```
DROP DATABASE|SCHEMA [IF EXISTS] 'DB_NAME';
```

范例：

```
12:05:30(root@localhost) [(none)]> drop database db1;
Query OK, 0 rows affected (0.01 sec)
```

```
12:06:41(root@localhost) [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| db2                |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| zabbix             |
+--------------------+
6 rows in set (0.01 sec)
[12:07:11 root@centos8 ~]#tree /var/lib/mysql -d
/var/lib/mysql
├── db2
├── #innodb_temp
├── mysql
├── performance_schema
├── sys
└── zabbix
```

3.3.4 查看数据库列表

```
SHOW DATABASES;
```

范例：

```
12:07:58(root@localhost) [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| db2                |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| zabbix             |
+--------------------+
6 rows in set (0.00 sec)
```

3.4 数据类型

数据类型：

- 数据长什么样
- 数据需要多少空间来存放

数据类型

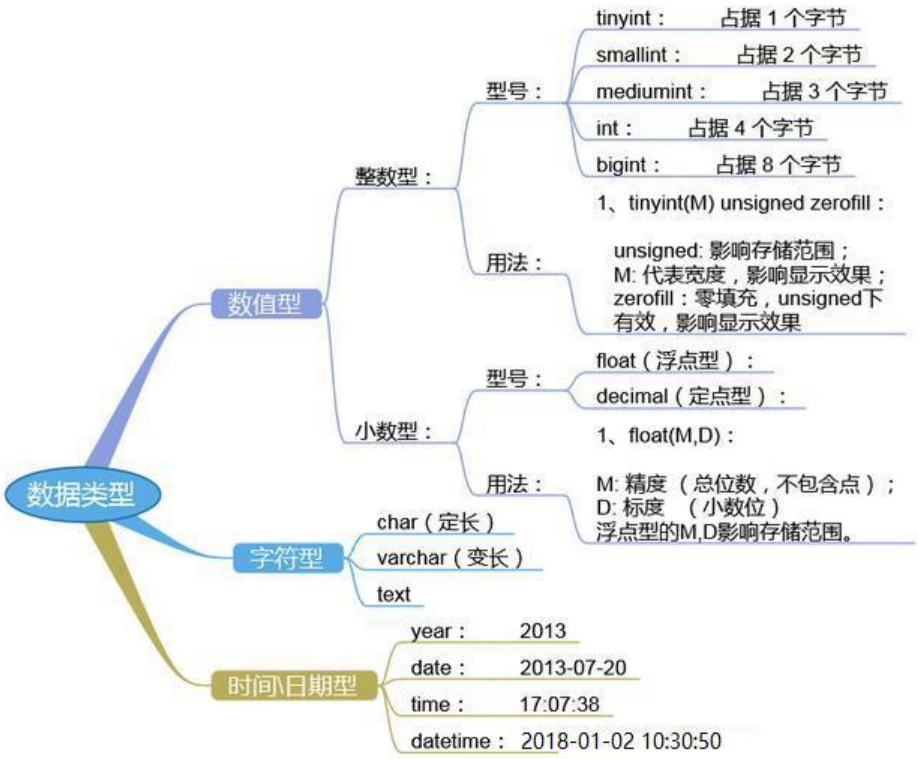
- 系统内置数据类型
- 用户定义数据类型

MySQL支持多种内置数据类型

- 数值类型
- 日期/时间类型
- 字符串(字符)类型

数据类型参考链接

https://dev.mysql.com/doc/refman/8.0/en/data-types.html



选择正确的数据类型对于获得高性能至关重要，三大原则：

1. 更小的通常更好，尽量使用可正确存储数据的最小数据类型
2. 简单就好，简单数据类型的操作通常需要更少的CPU周期
3. 尽量避免NULL，包含为NULL的列，对MySQL更难优化

3.4.1 整数型

tinyint(m) 1个字节 范围(-128~127)

smallint(m) 2个字节 范围(-32768~32767)

mediumint(m) 3个字节 范围(-8388608~8388607)

int(m) 4个字节 范围(-2147483648~2147483647)

bigint(m) 8个字节 范围(+-9.22*10的18次方)

上述数据类型，如果加修饰符unsigned后，则最大值翻倍

如：tinyint unsigned的取值范围为(0~255)

int(m)里的m是表示SELECT查询结果集中的显示宽度，并不影响实际的取值范围，规定了MySQL的些交互工具（例如MySQL命令行客户端）用来显示字符的个数。对于存储和计算来说，Int(1)和Int(20是相同的

BOOL，BOOLEAN：布尔型，是TINYINT(1)的同义词。zero值被视为假，非zero值视为真

3.4.2 浮点型(float和double)，近似值

float(m,d) 单精度浮点型 8位精度(4字节) m总个数，d小数位

double(m,d) 双精度浮点型16位精度(8字节) m总个数，d小数位

设一个字段定义为float(6,3)，如果插入一个数123.45678,实际数据库里存的是123.457，但总个数还实际为准，即6位

3.4.3 定点数

在数据库中存放的是精确值,存为十进制

decimal(m,d) 参数m<65 是总个数，d<30且 d<m 是小数位

MySQL5.0和更高版本将数字打包保存到一个二进制字符串中（每4个字节存9个数字）。

例如: decimal(18,9)小数点两边将各存储9个数字，一共使用9个字节：其中，小数点前的9个数字用4字节，小数点后的9个数字用4个字节，小数点本身占1个字节

浮点类型在存储同样范围的值时，通常比decimal使用更少的空间。float使用4个字节存储。double用8个字节

因为需要额外的空间和计算开销，所以应该尽量只在对小数进行精确计算时才使用decimal，例如存财务数据。但在数据量比较大的时候，可以考虑使用bigint代替decimal

3.4.4 字符串(char,varchar,text)

char(n) 固定长度，最多255个字符,注意不是字节

varchar(n) 可变长度，最多65535个字符

tinytext 可变长度，最多255个字符

text 可变长度，最多65535个字符

mediumtext 可变长度，最多2的24次方-1个字符

longtext 可变长度，最多2的32次方-1个字符

BINARY(M) 固定长度，可存二进制或字符，长度为0-M字节

VARBINARY(M) 可变长度，可存二进制或字符，允许长度为0-M字节

内建类型：ENUM枚举, SET集合

char和varchar的比较：

参考：https://dev.mysql.com/doc/refman/8.0/en/char.html

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

1. char(n) 若存入字符数小于n，则以空格补于其后，查询之时再将空格去掉，所以char类型存储的符串末尾不能有空格，varchar不限于此

2. char(n) 固定长度，char(4)不管是存入几个字符，都将占用4个字节，varchar是存入的实际字符数1个字节（n< n>255），所以varchar(4),存入3个字符将占用4个字节

3. char类型的字符串检索速度要比varchar类型的快

varchar 和 text：

1. varchar可指定n，text不能指定，内部存储varchar是存入的实际字符数+1个字节（n< n>255），txt是实际字符数+2个字节。

2. text类型不能有默认值

3. varchar可直接创建索引，text创建索引要指定前多少个字符。varchar查询速度快于text

3.4.5 二进制数据BLOB

BLOB和text存储方式不同，TEXT以文本方式存储，英文存储区分大小写，而Blob以二进制方式存储不分大小写

BLOB存储的数据只能整体读出

TEXT可以指定字符集，BLOB不用指定字符集

3.4.6 日期时间类型

date 日期 '2008-12-2'

time 时间 '12:25:36'

datetime 日期时间 '2008-12-2 22:06:44'

timestamp 自动存储记录修改时间

YEAR(2), YEAR(4)：年份

timestamp字段里的时间数据会随其他字段修改的时候自动刷新，这个数据类型的字段可以存放这条录最后被修改的时间

3.4.7 修饰符

NULL 数据列可包含NULL值，默认值

NOT NULL 数据列不允许包含NULL值，相当于网站注册表中的 * 为必填选项

DEFAULT 默认值

PRIMARY KEY 主键，所有记录中此字段的值不能重复，且不能为NULL

UNIQUE KEY 唯一键，所有记录中此字段的值不能重复，但可以为NULL

CHARACTER SET name 指定一个字符集

适用数值型的修饰符：

AUTO_INCREMENT 自动递增，适用于整数类型

UNSIGNED 无符号

范例：AUTO_INCREMENT

```
02:27:31(root@localhost) [(none)]> create database test;
Query OK, 1 row affected (0.01 sec)

02:27:46(root@localhost) [(none)]> use test
Database changed
02:27:54(root@localhost) [test]> create table t1(id int unsigned auto_increment primary key)
uto_increment = 4294967294;
Query OK, 0 rows affected (0.02 sec)
02:31:43(root@localhost) [test]> show table status from test like "t1" \G
*************************** 1. row ***************************
Name: t1
Engine: InnoDB
Version: 10
Row_format: Dynamic
Rows: 0
Avg_row_length: 0
Data_length: 16384
Max_data_length: 0
Index_length: 0
Data_free: 0
Auto_increment: 4294967294
Create_time: 2021-01-31 14:29:48
Update_time: NULL
Check_time: NULL
Collation: utf8_general_ci
Checksum: NULL
Create_options:
Comment:
1 row in set (0.00 sec)

02:33:15(root@localhost) [test]> insert into t1 values(null);
Query OK, 1 row affected (0.00 sec)

02:33:19(root@localhost) [test]> select * from t1;
+------------+
```

```
| id         |
+------------+
| 4294967294 |
| 4294967295 |
+------------+
2 rows in set (0.00 sec)

02:33:32(root@localhost) [test]> insert into t1 values(null);
ERROR 1062 (23000): Duplicate entry '4294967295' for key 't1.PRIMARY'
#上面表的数据类型无法存放所有数据,修改过数据类型实现自增长数据的增加
02:33:42(root@localhost) [test]> alter table t1 modify id bigint auto_increment;
Query OK, 2 rows affected (0.03 sec)
Records: 2  Duplicates: 0  Warnings: 0

02:35:02(root@localhost) [test]> desc t1;
+-------+--------+------+-----+---------+----------------+
| Field | Type   | Null | Key | Default | Extra          |
+-------+--------+------+-----+---------+----------------+
| id    | bigint | NO   | PRI | NULL    | auto_increment |
+-------+--------+------+-----+---------+----------------+
1 row in set (0.01 sec)

02:35:12(root@localhost) [test]> insert t1 values(null);
Query OK, 1 row affected (0.00 sec)

02:35:34(root@localhost) [test]> select * from t1;
+------------+
| id         |
+------------+
| 4294967294 |
| 4294967295 |
| 4294967296 |
+------------+
3 rows in set (0.00 sec)
```

3.5 DDL 语句

表：二维关系

设计表：遵循规范

定义：字段，索引

- 字段：字段名，字段数据类型，修饰符
- 约束，索引：应该创建在经常用作查询条件的字段上

3.5.1 创建表

创建表：

CREATE TABLE

获取帮助：

```
HELP CREATE TABLE
```

创建表的方法

```
CREATE TABLE [IF NOT EXISTS] 'tbl_name' (col1 type1 修饰符, col2 type2 修饰符,
...)
#字段信息
col type1
PRIMARY KEY(col1,...)
INDEX(col1, ...)
UNIQUE KEY(col1, ...)
#表选项:
ENGINE [=] engine_name
ROW_FORMAT [=] {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}
```

注意:

- Storage Engine是指表类型，也即在表创建时指明其使用的存储引擎
- 同一库中不同表可以使用不同的存储引擎
- 同一个库中表建议要使用同一种存储引擎类型

范例：创建表

```
2:46:55(root@localhost) [db1]> create table student ( id int unsigned auto_increment primary
key, name varchar(20) not null, age tinyint unsigned, gender enum('M','F') default 'M' )ENGIN
=InnoDB auto_increment=10 default charset=utf8;
Query OK, 0 rows affected, 1 warning (0.02 sec)
#id字段以10初始值

02:46:57(root@localhost) [db1]> desc student;
+--------+------------------+------+-----+---------+----------------+
| Field  | Type             | Null | Key | Default | Extra          |
+--------+------------------+------+-----+---------+----------------+
| id     | int unsigned     | NO   | PRI | NULL    | auto_increment |
| name   | varchar(20)      | NO   |     | NULL    |                |
| age    | tinyint unsigned | YES  |     | NULL    |                |
| gender | enum('M','F')    | YES  |     | M       |                |
+--------+------------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

02:48:08(root@localhost) [db1]> insert student (name,age) values('xiaoming',20);
Query OK, 1 row affected (0.00 sec)

02:48:20(root@localhost) [db1]> select * from student;
+----+----------+------+--------+
| id | name     | age  | gender |
+----+----------+------+--------+
| 10 | xiaoming |   20 | M      |
+----+----------+------+--------+
1 row in set (0.01 sec)
02:48:35(root@localhost) [db1]> insert student (name,age,gender) values('xiaohong',18,'f');
Query OK, 1 row affected (0.00 sec)
```

```
02:49:35(root@localhost) [db1]> select *from student;
+----+----------+------+--------+
| id | name     | age  | gender |
+----+----------+------+--------+
| 10 | xiaoming |   20 | M      |
| 11 | xiaohong |   18 | F      |
+----+----------+------+--------+
2 rows in set (0.00 sec)
```

范例：auto_increment 属性

```
02:49:49(root@localhost) [db1]> show variables like 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 1     |
| auto_increment_offset    | 1     |
+--------------------------+-------+
2 rows in set (0.01 sec)
#修改默认起始数
02:50:47(root@localhost) [db1]> set @@auto_increment_increment=10;
Query OK, 0 rows affected (0.00 sec)
#修改默认步长
02:51:34(root@localhost) [db1]> set @@auto_increment_offset=3;
Query OK, 0 rows affected (0.00 sec)

02:51:46(root@localhost) [db1]> show variables like 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 10    |
| auto_increment_offset    | 3     |
+--------------------------+-------+
2 rows in set (0.00 sec)
```

范例：时间类型

```
02:53:11(root@localhost) [db1]> create table testdate (id int auto_increment primary key,date
timestamp default current_timestamp not null);
Query OK, 0 rows affected (0.02 sec)
02:55:42(root@localhost) [db1]> insert testdate () values();
Query OK, 1 row affected (0.00 sec)

02:55:49(root@localhost) [db1]> insert testdate () values();
Query OK, 1 row affected (0.00 sec)

02:55:50(root@localhost) [db1]> insert testdate () values();
Query OK, 1 row affected (0.00 sec)

02:55:51(root@localhost) [db1]> select * from testdate;
+----+---------------------+
| id | date                |
+----+---------------------+
|  1 | 2021-01-31 14:55:49 |
```

```
|  2 | 2021-01-31 14:55:50 |
|  3 | 2021-01-31 14:55:51 |
+----+---------------------+
3 rows in set (0.00 sec)
```

(2) 通过查询现存表创建；新表会被直接插入查询而来的数据

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

范例：依据别的表创建新表并且把数据也插入，用于备份

```
02:58:57(root@localhost) [db1]> create table user select user,host from mysql.user;
Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0

03:01:53(root@localhost) [db1]> show tables;
+---------------+
| Tables_in_db1 |
+---------------+
| student       |
| testdate      |
| user          |
+---------------+
3 rows in set (0.00 sec)

03:02:04(root@localhost) [db1]> desc user;
+-------+-----------+------+-----+---------+-------+
| Field | Type      | Null | Key | Default | Extra |
+-------+-----------+------+-----+---------+-------+
| user  | char(32)  | NO   |     |         |       |
| host  | char(255) | NO   |     |         |       |
+-------+-----------+------+-----+---------+-------+
2 rows in set (0.00 sec)

03:02:10(root@localhost) [db1]> select * from user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
4 rows in set (0.00 sec)
```

(3) 通过复制现存的表的表结构创建，但不复制数据

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name { LIKE old_tbl_name | (LIKE
old_tbl_name) }
```

范例：只复制表结构

```
03:02:22(root@localhost) [db1]> desc student;
```

```
+--------+-----------------+------+-----+---------+----------------+
| Field  | Type            | Null | Key | Default | Extra          |
+--------+-----------------+------+-----+---------+----------------+
| id     | int unsigned    | NO   | PRI | NULL    | auto_increment |
| name   | varchar(20)     | NO   |     | NULL    |                |
| age    | tinyint unsigned | YES  |     | NULL    |                |
| gender | enum('M','F')   | YES  |     | M       |                |
+--------+-----------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

03:04:00(root@localhost) [db1]> create table teacher like student;
Query OK, 0 rows affected (0.02 sec)

03:04:21(root@localhost) [db1]> desc teacher;
+--------+-----------------+------+-----+---------+----------------+
| Field  | Type            | Null | Key | Default | Extra          |
+--------+-----------------+------+-----+---------+----------------+
| id     | int unsigned    | NO   | PRI | NULL    | auto_increment |
| name   | varchar(20)     | NO   |     | NULL    |                |
| age    | tinyint unsigned | YES  |     | NULL    |                |
| gender | enum('M','F')   | YES  |     | M       |                |
+--------+-----------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

3.5.2 表查看

查看表：

```
SHOW TABLES [FROM db_name]
```

查看表创建命令：

```
SHOW CREATE TABLE tbl_name
```

查看表结构：

```
DESC [db_name.]tb_name
SHOW COLUMNS FROM [db_name.]tb_name
```

查看表状态：

```
SHOW TABLE STATUS LIKE 'tbl_name'
```

查看支持的engine类型

```
SHOW ENGINES;
```

范例：查看表当前数据库表列表

```
03:04:28(root@localhost) [db1]> show tables;
+---------------+
| Tables_in_db1 |
+---------------+
| student       |
```

```
| teacher        |
| testdate       |
| user           |
+----------------+
4 rows in set (0.00 sec)
```

范例：查看表结构

```
03:07:32(root@localhost) [db1]> desc user;
+-------+-----------+------+-----+---------+-------+
| Field | Type      | Null | Key | Default | Extra |
+-------+-----------+------+-----+---------+-------+
| user  | char(32)  | NO   |     |         |       |
| host  | char(255) | NO   |     |         |       |
+-------+-----------+------+-----+---------+-------+
2 rows in set (0.00 sec)
03:07:43(root@localhost) [db1]> show columns from user;
+-------+-----------+------+-----+---------+-------+
| Field | Type      | Null | Key | Default | Extra |
+-------+-----------+------+-----+---------+-------+
| user  | char(32)  | NO   |     |         |       |
| host  | char(255) | NO   |     |         |       |
+-------+-----------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

范例：查看创建表命令

```
03:09:02(root@localhost) [db1]> show create table student;
+---------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Table   | Create Table                                                                                                                                                                                                                                        |
+---------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| student | CREATE TABLE `student` (
`id` int unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(20) NOT NULL,
`age` tinyint unsigned DEFAULT NULL,
`gender` enum('M','F') DEFAULT 'M',
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8 |
+---------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

范例：查看表状态

```
03:09:06(root@localhost) [db1]> show table status like 'student'\G
*************************** 1. row ***************************
Name: student
Engine: InnoDB
Version: 10
```

```
Row_format: Dynamic
Rows: 2
Avg_row_length: 8192
Data_length: 16384
Max_data_length: 0
Index_length: 0
Data_free: 0
Auto_increment: 12
Create_time: 2021-01-31 14:46:57
Update_time: 2021-01-31 14:49:35
Check_time: NULL
Collation: utf8_general_ci
Checksum: NULL
Create_options:
Comment:
1 row in set (0.01 sec)
```

查看库中所有表状态

```
SHOW TABLE STATUS FROM db_name
```

范例：

```
03:09:56(root@localhost) [db1]> show table status from db1\G
```

3.5.3 修改和删除表

修改表

```
ALTER TABLE 'tbl_name'
#字段：
#添加字段：add
ADD col1 data_type [FIRST|AFTER col_name]
#删除字段：drop
#修改字段：
alter（默认值），change（字段名），modify（字段属性）
```

查看修改表帮助

```
Help ALTER TABLE
```

删除表

```
DROP TABLE [IF EXISTS] 'tbl_name';
```

修改表范例

```
ALTER TABLE students RENAME s1;
ALTER TABLE s1 ADD phone varchar(11) AFTER name;
ALTER TABLE s1 MODIFY phone int;
ALTER TABLE s1 CHANGE COLUMN phone mobile char(11);
ALTER TABLE s1 DROP COLUMN mobile;
ALTER TABLE s1 character set utf8;
ALTER TABLE s1 change name name varchar(20) character set utf8;
```

```
ALTER TABLE students ADD gender ENUM('m','f');
ALETR TABLE students CHANGE id sid int UNSIGNED NOT NULL PRIMARY KEY;
ALTER TABLE students DROP age;
DESC students;

#新建表无主键，添加和删除主键
CREATE TABLE t1 SELECT * FROM students;
ALTER TABLE t1 add primary key (stuid);
ALTER TABLE t1 drop primary key ;
```

3.6 DML 语句

DML: INSERT, DELETE, UPDATE

3.6.1 INSERT 语句

功能：一次插入一行或多行数据

语法

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE #如果重复更新之
col_name=expr
[, col_name=expr] ... ]
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name
SET col_name={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
col_name=expr
[, col_name=expr] ... ]
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
SELECT ...
[ ON DUPLICATE KEY UPDATE
col_name=expr
[, col_name=expr] ... ]
```

简化写法：

```
INSERT tbl_name [(col1,...)] VALUES (val1,...), (val21,...)
```

3.6.2 UPDATE 语句

语法：

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

注意：一定要有限制条件，否则将修改所有行的指定字段

可利用mysql 选项避免此错误：

```
mysql -U | --safe-updates| --i-am-a-dummy
[root@centos8 ~]#vim /etc/my.cnf
[mysql]
safe-updates
```

4.6.3 DELETE 语句

删除表中数据，但不会自动缩减数据文件的大小。

语法:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
#可先排序再指定删除的行数
```

注意：一定要有限制条件，否则将清空表中的所有数据

如果想清空表，保留表结构，也可以使用下面语句，此语句会自动缩减数据文件的大小。

```
TRUNCATE TABLE tbl_name;
```

缩减表大小

```
OPTIMIZE TABLE tb_name
```