注释方式：

- -- <注释>
    - 注意有个空格
- # <注释>
- /* <注释> */

数据定义

创建、删除、查看数据库

- 创建数据库

```
-- 语法
CREATE DATABASE <database_name>;
# 举例
CREATE DATABASE mysql_0;
```

```
mysql> CREATE DATABASE mysql_0;
Query OK, 1 row affected (0.13 sec)
```

- 查看数据库

```
# 语法
SHOW DATABASES;
```

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysql_0            |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
7 rows in set (0.00 sec)
```

- 删除数据库

```
# 语法
DROP DATABASE <database_name>;
# 举例
DROP DATABASE mysql_0;
```

```
mysql> DROP DATABASE mysql_0;
Query OK, 0 rows affected (0.19 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
6 rows in set (0.00 sec)
```

- 进入数据库

```
# 语法
USE <database_name>;
# 举例
USE mysql_0;
```

```
mysql> USE mysql_0;
Database changed
```

创建、修改、删除基本表

- 创建基本表

```
# 语法
CREATE TABLE <table_name>
(
列名1 数据类型1 [列级完整性约束条件1],
列名2 数据类型2 [列级完整性约束条件1],
...
[表级完整性约束条件]
);
# 举例
CREATE TABLE C
(
CNO char(5) NOT NULL,
CN varchar(20),
CT int,
PRIMARY KEY (CNO),
CHECK (CT >= 1)
) default charset utf8; # 如果文本编码没有问题，最后一行可以省略，只保留右括号与分号。

CREATE TABLE S
(
SNO char(5),
SN varchar(8) NOT NULL,
SEX char(2) NOT NULL,
AGE int NOT NULL,
DEPT varchar(8),
PRIMARY KEY (SNO),
UNIQUE (SN),
CONSTRAINT ck_S CHECK (SEX IN ('男','女') AND AGE > 0)
) default charset utf8;

CREATE TABLE SC
(
SNO char(5) NOT NULL,
CNO char(5) NOT NULL,
SCORE numeric(3, 0),
CONSTRAINT pk_SC PRIMARY KEY (SNO, CNO),
CONSTRAINT fk_SNO_SC FOREIGN KEY (SNO) REFERENCES S(SNO),
CONSTRAINT fk_CNO_SC FOREIGN KEY (CNO) REFERENCES C(CNO)
```

```
) default charset utf8;
```

```
mysql> CREATE TABLE C
    -> (
    -> CNO char(5) NOT NULL,
    -> CN varchar(20),
    -> CT int,
    -> PRIMARY KEY (CNO),
    -> CHECK (CT >= 1)
    -> ) default charset utf8;
Query OK, 0 rows affected, 1 warning (0.87 sec)

mysql> CREATE TABLE S
    -> (
    -> SNO char(5),
    -> SN varchar(8) NOT NULL,
    -> SEX char(2) NOT NULL,
    -> AGE int NOT NULL,
    -> DEPT varchar(8),
    -> PRIMARY KEY (SNO),
    -> UNIQUE (SN),
    -> CONSTRAINT ck_S CHECK (SEX IN ('男','女') AND AGE > 0)
    -> ) default charset utf8;
Query OK, 0 rows affected, 1 warning (0.71 sec)

mysql> CREATE TABLE SC
    -> (
    -> SNO char(5) NOT NULL,
    -> CNO char(5) NOT NULL,
    -> SCORE numeric(3, 0),
    -> CONSTRAINT pk_SC PRIMARY KEY (SNO, CNO),
    -> CONSTRAINT fk_SNO_SC FOREIGN KEY (SNO) REFERENCES S(SNO),
    -> CONSTRAINT fk_CNO_SC FOREIGN KEY (CNO) REFERENCES C(CNO)
    -> ) default charset utf8;
Query OK, 0 rows affected, 1 warning (0.66 sec)
```

- MySQL数据类型
  - 数值型

数据类型	描述
TINYINT(size)	带符号-128到127 ，无符号0到255。
SMALLINT(size)	带符号范围-32768到32767，无符号0到65535, size 默认为 6。
MEDIUMINT(size)	带符号范围-8388608到8388607，无符号的范围是0到16777215。 size 默认为9
INT(size)	带符号范围-2147483648到2147483647，无符号的范围是0到4294967295。 size 默认为 11
BIGINT(size)	带符号的范围是-9223372036854775808到9223372036854775807，无符号的范围是0到18446744073709551615。size 默认为 20
FLOAT(size,d)	带有浮动小数点的小数字。在 size 参数中规定显示最大位数。在 d 参数中规定小数点右侧的最大位数。
DOUBLE(size,d)	带有浮动小数点的大数字。在 size 参数中规显示定最大位数。在 d 参数中规定小数点右侧的最大位数。
DECIMAL(size,d)	作为字符串存储的 DOUBLE 类型，允许固定的小数点。在 size 参数中规定显示最大位数。在 d 参数中规定小数点右侧的最大位数。

- 文本型

数据类型	描述
CHAR(size)	保存**固定长度**的字符串（可包含字母、数字以及特殊字符）。在括号中指定字符串的长度。最多 255 个字符。
VARCHAR(size)	保存**可变长度**的字符串（可包含字母、数字以及特殊字符）。在括号中指定字符串的最大长度。最多 255 个字符。**注释：**如果值的长度大于 255，则被转换为 TEXT 类型。
TINYTEXT	存放最大长度为 255 个字符的字符串。
TEXT	存放最大长度为 65,535 个字符的字符串。
BLOB	用于 BLOBs（Binary Large OBjects）。存放最多 65,535 字节的数据。
MEDIUMTEXT	存放最大长度为 16,777,215 个字符的字符串。
MEDIUMBLOB	用于 BLOBs（Binary Large OBjects）。存放最多 16,777,215 字节的数据。
LONGTEXT	存放最大长度为 4,294,967,295 个字符的字符串。
LONGBLOB	用于 BLOBs (Binary Large OBjects)。存放最多 4,294,967,295 字节的数据。
ENUM(x,y,z,etc.)	允许您输入可能值的列表。可以在 ENUM 列表中列出最大 65535 个值。如果列表中不存在插入的值，则插入空值。**注释：**这些值是按照您输入的顺序排序的。可以按照此格式输入可能的值： ENUM('X','Y','Z')
SET	与 ENUM 类似，不同的是，SET 最多只能包含 64 个列表项且 SET 可存储一个以上的选择。

- 日期型

数据类型	描述
DATE()	日期。格式：YYYY-MM-DD**注释：**支持的范围是从 '1000-01-01' 到 '9999-12-31'
DATETIME()	*日期和时间的组合。格式：YYYY-MM-DD HH:MM:SS**注释：**支持的范围是从 '1000-01-01 00:00:00' 到 '9999-12-31 23:59:59'
TIMESTAMP()	*时间戳。TIMESTAMP 值使用 Unix 纪元('1970-01-01 00:00:00' UTC) 至今的秒数来存储。格式：YYYY-MM-DD HH:MM:SS**注释：**支持的范围是从 '1970-01-01 00:00:01' UTC 到 '2038-01-09 03:14:07' UTC
TIME()	时间。格式：HH:MM:SS**注释：**支持的范围是从 '-838:59:59' 到 '838:59:59'
YEAR()	2 位或 4 位格式的年。**注释：**4 位格式所允许的值：1901 到 2155。2 位格式所允许的值：70 到 69，表示从 1970 到 2069。

- SQL约束条件
  - NULL / NOT NULL

空值不等于0也不等于空白，而是表示不知道、不确定、没有数据的意思，该约束中用于列约束。在默认的情况下，表的列接受NULL值。

语法：

```
[CONSTRAINT <约束名>][NULL/NOT NULL]
```

  - UNIQUE

UNIQUE 约束表示某一列或多个列的组合上的取值必须唯一，系统会自动为其建立唯一索引。它既可以用于列约束，也可以用于表约束。

语法：

```
[CONSTRAINT <约束名>]UNIQUE[CLUSTERD/NONCLUSTERED]
[(column_name[ASC/DESC])]
```

ASC:按升序建立索引

DESC:按降序建立索引

  - PRIMARY KEY

PRIMARY KEY 约束唯一标识数据库表中的每条记录。主键必须包含唯一的值。

主键列不能包含 NULL 值。每个表都应该有一个主键，并且每个表只能有一个主键。它既可以用于列约束，也可以用于表约束。

定义列约束时的语法：

```
[CONSTRAINT <约束名>]PRIMARY KEY[CLUSTERED/NONCLUSTERED]
[(column_name[ASC/DESC])]
```

  - FOREIGN KEY

FOREIGN KEY约束指定某一列或一组列作为外部键，其中，包含外部键的表称为从表，包含外部键引用的主键或唯一键称为主表。系统保证从表在外部键上的取值是主表中某一个主键或唯一键值，或者取空值，以此来保证两个表的连接。FOREIGN KEY约束用于预防破坏表之间连接的行为，能防止非法数据插入外键列。它既可以用于列约束，也可以用于表约束。

语法：

```
[CONSTRAINT <约束名>]FOREIGN KEY REFERENCES <主表名>(<主表列名>)

[CONSTRAINT <约束名>]FOREIGN KEY [(从表列名)] REFERENCES <主表名>[(<主表列名>)]
```

- CHECK

CHECK 约束可定义用户自定义的完整性约束规则。它既可以用于列约束，也可以用于表约束。如果对单个列定义CHECK约束，那么该列只允许特定的值。

语法：

```
[CONSTRAINT <约束名>]CHECK[NOT FOR REPLICATION](<条件>)
```

- DEFAULT

DEFAULT 约束用于向列中插入默认值。如果没有规定其它的值，那么会将默认值添加到所有的新纪录。

语法：

```
[CONSTRAINT <约束名>]DEFAULT <默认值>
```

- 查看基本表

```
# 查看当前数据库中所有的表
    # 语法
SHOW TABLES;
# 查看某个表的结构
    # 语法
DESC table_name;
# 举例
DESC C;
```

```
mysql> SHOW TABLES;
+-------------------+
| Tables_in_mysql_0 |
+-------------------+
| c                 |
| s                 |
| sc                |
+-------------------+
3 rows in set (0.00 sec)

mysql> DESC C;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| CNO   | char(5)     | NO   | PRI | NULL    |       |
| CN    | varchar(20) | YES  |     | NULL    |       |
| CT    | int(11)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- 修改基本表

```
# SQL语言用ALTER TABLE命令来修改基本表。修改基本表的行为包含：添加新列，添加列的完整性约束条件，删除列，删除列的完整性约束条件，改变原有列的数据类型，禁用参照完整性。
# 语法

```

```
ALTER TABLE <表名>
(
    ALTER COLUMN 列名 新的数据类型, # 修改表中指定列的数据类型，可能会与原数据类型不兼容而报错
    ADD 新列名 数据类型, # 新增一列
    ADD 列名 完整性约束条件, # 修改表中指定列的约束条件
    DROP COLUMN 列名, # 删除指定列
    DROP CONSTRAINT 约束名, # 删除指定列的约束条件
    CHECK/NOCHECK CONSTRAINT 约束名, # 启动/禁用指定的约束条件
    ENABLE/DISABLE TRIGGER 触发器名 # 启动/禁用指定的触发器
);
```

修改表后可以查看表的结构是否改变。

- 删除基本表

```
# 语法
DROP TABLE <table_name>;
# 注意
    # 删除表需要相应的操作权限，一般只删除自己建立的无用表；执行删除命令后是否真能完成删除操作，取决与其操作是否违反了完整性约束条件。

```

```
mysql> DROP TABLE SC;
Query OK, 0 rows affected (0.43 sec)

mysql> SHOW TABLES;
+------------------+
| Tables_in_mysql_0 |
+------------------+
| c                 |
| s                 |
+------------------+
2 rows in set (0.05 sec)
```

修改表名

```
# 语法
ALTER TABLE old_name RENAME TO new_name;
# 注意
    # 表名、视图名、列名都可以修改，但是数据库名是无法修改的。
```

创建、删除索引

- 索引的概念

  索引是双刃剑，添加索引，可以加速数据查询，但会减慢更新速度。

  索引分为聚集索引和非聚集索引，聚集索引比非聚集索引速度快，一个表中只有一个聚集索引，但可以有多个非聚集索引。

聚集索引一般是表中的主键索引，如果表中没有显示指定主键，则会选择表中的第一个不允许为NULL的唯一索引，如果还是没有的话，就采用 Innodb 存储引擎为每行数据内置的6字节 ROWID 作为聚集索引。

- 创建索引

```
# 语法
CREATE  [UNIQUE]
[CLUSTERED/NONCLUSTERED]
INDEX<索引名>ON{<表名>/<视图名>}(<列名>[ASC/DESC])
    # 其中，UNIQUE表示唯一索引，CLUSTERED表示建立聚集索引，NONCLUSTERED表示建立非聚集索引。
# 注意
    # MySQL的语法定义聚集索引与非聚集索引与上述语法稍有不同，不能用上面的语法定义聚集/非聚集索引，会报错。
```

- 删除索引

```
# 语法
DROP INDEX table_name<index_name>;
DROP INDEX view_name<index_name>;
# MySQL 语法
ALTER TABLE table_name DROP INDEX index_name;
```

数据查询

SELECT

SELECT 语句用于从数据库中选取数据，结果被存储在一个结果表中，称为结果集。

```
# 语法 MySQL
SELECT column_name1,column_name2 FROM table_name
[WHERE exp]
[LIMIT N][OFFSET M]
    # 注意
-- * 查询语句中你可以使用一个或者多个表，表之间使用逗号(,)分割，并使用WHERE语句来设定查询条件。
-- * SELECT 命令可以读取一条或者多条记录。
-- * 使用星号（*）来代替其他字段，SELECT语句会返回表的所有字段数据
-- * 使用 WHERE 语句来包含任何条件。
-- * 使用 LIMIT 属性来设定返回的记录数。
-- * 通过OFFSET指定SELECT语句开始查询的数据偏移量。默认情况下偏移量为0。
```

```
# 语法 SQL Server
SELECT [ALL/DISTINCT] <目标表达式1>[[AS] 列别名1][,<目标表达式2>[[AS] 列别名2]]...
[INTO <新表名>]
FROM <表名1/视图名1>[[AS] 表别名1][,<表名2/视图名2>[[AS] 表别名2]]...
[WHERE <元组/记录筛选条件表达式>]
[GROUP BY <列名11>[,<列名12>]...[HAVING <分组筛选条件表达式>]]
[ORDER BY <列名21>[ASC/DESC][,<列名22>[ASC/DESC]]...]
```

查询整表

```
mysql> SELECT * FROM C;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |    4 |
| C2  | 离散数学 |    2 |
| C3  | 操作系统 |    3 |
| C4  | 数据结构 |    4 |
| C5  | 数据库   |    4 |
| C6  | 汇编语言 |    3 |
| C7  | 信息基础 |    2 |
+-----+----------+------+
7 rows in set (0.08 sec)
```

查询某列，某些列

```
mysql> SELECT CN FROM C;
+----------+
| CN       |
+----------+
| C语言    |
| 离散数学 |
| 操作系统 |
| 数据结构 |
| 数据库   |
| 汇编语言 |
| 信息基础 |
+----------+
7 rows in set (0.00 sec)

mysql> SELECT CNO,CN FROM C;
+-----+----------+
| CNO | CN       |
+-----+----------+
| C1  | C语言    |
| C2  | 离散数学 |
| C3  | 操作系统 |
| C4  | 数据结构 |
| C5  | 数据库   |
| C6  | 汇编语言 |
| C7  | 信息基础 |
+-----+----------+
7 rows in set (0.00 sec)
```

添加表达式与文本列

```
mysql> SELECT SN,2019-AGE FROM S;
+------+----------+
| SN   | 2019-AGE |
+------+----------+
| 李   |     1998 |
| 王   |     1999 |
| 陈   |     1996 |
| 张   |     2000 |
| 吴   |     1998 |
```

```
| 徐   |     1997 |
| 陈东 |     1999 |
+------+----------+
7 rows in set (0.00 sec)

mysql> SELECT SN,'出生日期',2019-AGE,lower(DEPT) FROM S;
+------+----------+----------+-------------+
| SN   | 出生日期 | 2019-AGE | lower(DEPT) |
+------+----------+----------+-------------+
| 李   | 出生日期 |     1998 | 信息        |
| 王   | 出生日期 |     1999 | 计算机      |
| 陈   | 出生日期 |     1996 | 自动化      |
| 张   | 出生日期 |     2000 | 自动化      |
| 吴   | 出生日期 |     1998 | 信息        |
| 徐   | 出生日期 |     1997 | 计算机      |
| 陈东 | 出生日期 |     1999 | 信息        |
+------+----------+----------+-------------+
7 rows in set (0.00 sec)
```

替换列名

```
mysql> SELECT SN,'出生日期',2019-AGE,lower(DEPT) FROM S;
/* 同时显示新列名与旧列名
new_column_name(old_column_name)
*/
+------+----------+----------+-------------+
| SN   | 出生日期 | 2019-AGE | lower(DEPT) |
+------+----------+----------+-------------+
| 李   | 出生日期 |     1998 | 信息        |
| 王   | 出生日期 |     1999 | 计算机      |
| 陈   | 出生日期 |     1996 | 自动化      |
| 张   | 出生日期 |     2000 | 自动化      |
| 吴   | 出生日期 |     1998 | 信息        |
| 徐   | 出生日期 |     1997 | 计算机      |
| 陈东 | 出生日期 |     1999 | 信息        |
+------+----------+----------+-------------+
7 rows in set (0.00 sec)

mysql> SELECT SN SNAME,'出生日期:' BIRTH,2019-AGE BIRTHDAY,DEPT AS DEPARTMENT
FROM S;
/* 替换列名成新
-- old_column_name new_column_name
-- old_column_name ASnew_column_name
*/
+-------+-----------+----------+------------+
| SNAME | BIRTH     | BIRTHDAY | DEPARTMENT |
+-------+-----------+----------+------------+
| 李    | 出生日期: |     1998 | 信息       |
| 王    | 出生日期: |     1999 | 计算机     |
| 陈    | 出生日期: |     1996 | 自动化     |
| 张    | 出生日期: |     2000 | 自动化     |
| 吴    | 出生日期: |     1998 | 信息       |
| 徐    | 出生日期: |     1997 | 计算机     |
| 陈东  | 出生日期: |     1999 | 信息       |
+-------+-----------+----------+------------+
7 rows in set (0.00 sec)
```

去除重复项

使用 DISTINCT 选项

```
mysql> SELECT SNO FROM SC;
+-----+
| SNO |
+-----+
| S1  |
| S3  |
| S1  |
| S4  |
| S5  |
| S3  |
| S3  |
| S4  |
| S4  |
| S3  |
+-----+
10 rows in set (0.03 sec)

mysql> SELECT DISTINCT SNO FROM SC;
+-----+
| SNO |
+-----+
| S1  |
| S3  |
| S4  |
| S5  |
+-----+
4 rows in set (0.01 sec)
```

WHERE 语句

添加 WHERE 子句限定查询结果

```
mysql> SELECT SN FROM S;
+------+
| SN   |
+------+
| 吴   |
| 张   |
| 徐   |
| 李   |
| 王   |
| 陈   |
| 陈东 |
+------+
7 rows in set (0.01 sec)

mysql> SELECT SN FROM S WHERE DEPT='计算机';
-- WHERE 表达式可以使用等式，也可以使用 IS 或 NOT IS ，不在赘述
+----+
| SN |
+----+
| 王 |
```

```
| 徐 |
+----+
2 rows in set (0.00 sec)

mysql> SELECT SNO,AGE FROM S WHERE AGE < 20;
-- WHERE 表达式可以使用比较表达式
+-----+-----+
| SNO | AGE |
+-----+-----+
| S4  |  19 |
+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT SNO,AGE FROM S WHERE AGE BETWEEN 20 AND 30;
-- WHERE 表达式可以使用 BETWEEN ... AND ... 或加 NOT 句式
+-----+-----+
| SNO | AGE |
+-----+-----+
| S1  |  21 |
| S2  |  20 |
| S3  |  23 |
| S5  |  21 |
| S6  |  22 |
| S7  |  20 |
+-----+-----+
6 rows in set (0.00 sec)

mysql> SELECT SNO,AGE FROM S WHERE AGE NOT BETWEEN 20 AND 30;
+-----+-----+
| SNO | AGE |
+-----+-----+
| S4  |  19 |
+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT SNO,AGE FROM S WHERE AGE IN (21,20);
-- WHERE 表达式可以使用 IN 加 一个选项集合 的句式，也可以添加 NOT
+-----+-----+
| SNO | AGE |
+-----+-----+
| S1  |  21 |
| S2  |  20 |
| S5  |  21 |
| S7  |  20 |
+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT SNO,AGE FROM S WHERE AGE NOT IN (21,20);
+-----+-----+
| SNO | AGE |
+-----+-----+
| S3  |  23 |
| S4  |  19 |
| S6  |  22 |
+-----+-----+
3 rows in set (0.00 sec)

mysql> SELECT SNO,AGE FROM S WHERE AGE LIKE '%1';
```

```
-- WHERE 表达式可以使用 LIKE 加通配符表达式的形式。这里的通配符表达式需要使用单引号括起，_表示单一字符，%表示任意字符。如果在原列记录中包含这两个字符，为防止错误，要使用反义符 \ 。
+-----+-----+
| SNO | AGE |
+-----+-----+
| S1  |  21 |
| S5  |  21 |
+-----+-----+
2 rows in set (0.00 sec)

mysql> SELECT SNO,AGE FROM S WHERE AGE LIKE '_1';
+-----+-----+
| SNO | AGE |
+-----+-----+
| S1  |  21 |
| S5  |  21 |
+-----+-----+
2 rows in set (0.00 sec)
# 多个WHERE表达式可以使用 AND 或 OR 连接
```

统计函数

SELECT 语句可以添加一些统计函数进行统计汇总查询

```
mysql> SELECT COUNT(*) AS '学生总人数' FROM S;
+------------+
| 学生总人数 |
+------------+
|          7 |
+------------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(DISTINCT SNO) FROM SC;
+----------------------+
| COUNT(DISTINCT SNO) |
+----------------------+
|                    4 |
+----------------------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(SNO) FROM SC;
+------------+
| COUNT(SNO) |
+------------+
|         10 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM SC;
+-----+-----+-------+
| SNO | CNO | SCORE |
+-----+-----+-------+
| S1  | C1  |    90 |
| S1  | C2  |    85 |
| S3  | C1  |    73 |
```

```
| S3  | C4  |    88 |
| S3  | C5  |    85 |
| S3  | C7  |    68 |
| S4  | C2  |    65 |
| S4  | C5  |    90 |
| S4  | C6  |    79 |
| S5  | C2  |    89 |
+-----+-----+-------+
10 rows in set (0.00 sec)

mysql> SELECT COUNT(*),MAX(SCORE),MIN(SCORE),AVG(SCORE) FROM SC;
+----------+------------+------------+------------+
| COUNT(*) | MAX(SCORE) | MIN(SCORE) | AVG(SCORE) |
+----------+------------+------------+------------+
|       10 |         90 |         65 |    81.2000 |
+----------+------------+------------+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM SC;
+-----+-----+-------+
| SNO | CNO | SCORE |
+-----+-----+-------+
| S1  | C1  |    90 |
| S1  | C2  |    85 |
| S3  | C1  |    73 |
| S3  | C4  |    88 |
| S3  | C5  |    85 |
| S3  | C7  |    68 |
| S4  | C2  |    65 |
| S4  | C5  |    90 |
| S4  | C6  |    79 |
| S5  | C2  |    89 |
+-----+-----+-------+
10 rows in set (0.00 sec)

mysql> SELECT CNO,COUNT(SNO) AS '选课人数' FROM SC GROUP BY CNO;
+-----+----------+
| CNO | 选课人数 |
+-----+----------+
| C1  |        2 |
| C2  |        3 |
| C4  |        1 |
| C5  |        2 |
| C6  |        1 |
| C7  |        1 |
+-----+----------+
6 rows in set (0.00 sec)

mysql> SELECT CNO,COUNT(SNO) FROM SC GROUP BY CNO HAVING COUNT(*)>=2;
+-----+------------+
| CNO | COUNT(SNO) |
+-----+------------+
| C1  |          2 |
| C2  |          3 |
| C5  |          2 |
+-----+------------+
3 rows in set (0.00 sec)
```

查询的顺序

```
mysql> SELECT SNO,SCORE FROM SC WHERE CNO='C2' ORDER BY SCORE DESC;
+-----+-------+
| SNO | SCORE |
+-----+-------+
| S5  |    89 |
| S1  |    85 |
| S4  |    65 |
+-----+-------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM S ORDER BY DEPT,AGE DESC;
+-----+------+-----+-----+--------+
| SNO | SN   | SEX | AGE | DEPT   |
+-----+------+-----+-----+--------+
| S1  | 李   | 男  |  21 | 信息   |
| S5  | 吴   | 女  |  21 | 信息   |
| S7  | 陈东 | 男  |  20 | 信息   |
| S3  | 陈   | 女  |  23 | 自动化 |
| S4  | 张   | 男  |  19 | 自动化 |
| S6  | 徐   | 女  |  22 | 计算机 |
| S2  | 王   | 女  |  20 | 计算机 |
+-----+------+-----+-----+--------+
7 rows in set (0.00 sec)
```

连接查询

```
mysql> SELECT * FROM S,SC WHERE S.SNO=SC.SNO;
+-----+----+-----+-----+--------+-----+-----+-------+
| SNO | SN | SEX | AGE | DEPT   | SNO | CNO | SCORE |
+-----+----+-----+-----+--------+-----+-----+-------+
| S1  | 李 | 男  |  21 | 信息   | S1  | C1  |    90 |
| S1  | 李 | 男  |  21 | 信息   | S1  | C2  |    85 |
| S3  | 陈 | 女  |  23 | 自动化 | S3  | C1  |    73 |
| S3  | 陈 | 女  |  23 | 自动化 | S3  | C4  |    88 |
| S3  | 陈 | 女  |  23 | 自动化 | S3  | C5  |    85 |
| S3  | 陈 | 女  |  23 | 自动化 | S3  | C7  |    68 |
| S4  | 张 | 男  |  19 | 自动化 | S4  | C2  |    65 |
| S4  | 张 | 男  |  19 | 自动化 | S4  | C5  |    90 |
| S4  | 张 | 男  |  19 | 自动化 | S4  | C6  |    79 |
| S5  | 吴 | 女  |  21 | 信息   | S5  | C2  |    89 |
+-----+----+-----+-----+--------+-----+-----+-------+
10 rows in set (0.00 sec)

mysql> SELECT S.SNO,SN,SEX,AGE,DEPT,CNO,SCORE FROM S,SC WHERE S.SNO=SC.SNO;
+-----+----+-----+-----+--------+-----+-------+
| SNO | SN | SEX | AGE | DEPT   | CNO | SCORE |
+-----+----+-----+-----+--------+-----+-------+
| S1  | 李 | 男  |  21 | 信息   | C1  |    90 |
| S1  | 李 | 男  |  21 | 信息   | C2  |    85 |
| S3  | 陈 | 女  |  23 | 自动化 | C1  |    73 |
| S3  | 陈 | 女  |  23 | 自动化 | C4  |    88 |
| S3  | 陈 | 女  |  23 | 自动化 | C5  |    85 |
| S3  | 陈 | 女  |  23 | 自动化 | C7  |    68 |
| S4  | 张 | 男  |  19 | 自动化 | C2  |    65 |
```

```
| S4  | 张 | 男  |  19 | 自动化 | C5  |    90 |
| S4  | 张 | 男  |  19 | 自动化 | C6  |    79 |
| S5  | 吴 | 女  |  21 | 信息   | C2  |    89 |
+-----+----+-----+-----+--------+-----+-------+
10 rows in set (0.00 sec)

mysql> SELECT S.SNO,SN,SEX,AGE,DEPT,CNO,SCORE FROM S LEFT OUTER JOIN SC ON
S.SNO=SC.SNO;
+-----+------+-----+-----+--------+------+-------+
| SNO | SN   | SEX | AGE | DEPT   | CNO  | SCORE |
+-----+------+-----+-----+--------+------+-------+
| S1  | 李   | 男  |  21 | 信息   | C1   |    90 |
| S1  | 李   | 男  |  21 | 信息   | C2   |    85 |
| S2  | 王   | 女  |  20 | 计算机 | NULL |  NULL |
| S3  | 陈   | 女  |  23 | 自动化 | C1   |    73 |
| S3  | 陈   | 女  |  23 | 自动化 | C4   |    88 |
| S3  | 陈   | 女  |  23 | 自动化 | C5   |    85 |
| S3  | 陈   | 女  |  23 | 自动化 | C7   |    68 |
| S4  | 张   | 男  |  19 | 自动化 | C2   |    65 |
| S4  | 张   | 男  |  19 | 自动化 | C5   |    90 |
| S4  | 张   | 男  |  19 | 自动化 | C6   |    79 |
| S5  | 吴   | 女  |  21 | 信息   | C2   |    89 |
| S6  | 徐   | 女  |  22 | 计算机 | NULL |  NULL |
| S7  | 陈东 | 男  |  20 | 信息   | NULL |  NULL |
+-----+------+-----+-----+--------+------+-------+
13 rows in set (0.00 sec)
```

合并查询

```
SELECT SNO AS 学号,SUM(SCORE) AS 总分 FROM SC WHERE SNO='S1'
UNION
SELECT SNO AS 学号,SUM(SCORE) AS 总分 FROM SC WHERE SNO='S5';
```

```
mysql> SELECT SNO AS 学号,SUM(SCORE) AS 总分 FROM SC WHERE SNO='S1'
    -> UNION
    -> SELECT SNO AS 学号,SUM(SCORE) AS 总分 FROM SC WHERE SNO='S5';
+------+------+
| 学号 | 总分 |
+------+------+
| S1   |  175 |
| S5   |   89 |
+------+------+
2 rows in set (0.00 sec)
```

嵌套查询

```
mysql> SELECT SNO,SN,DEPT FROM S WHERE DEPT IN(SELECT DEPT FROM S WHERE
SN='王');
+-----+----+--------+
| SNO | SN | DEPT   |
+-----+----+--------+
| S2  | 王 | 计算机 |
| S6  | 徐 | 计算机 |
+-----+----+--------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT A.SNO,A.SN,B.DEPT FROM S A,S B WHERE A.DEPT=B.DEPT AND
B.SN='王';
+-----+----+--------+
| SNO | SN | DEPT   |
+-----+----+--------+
| S2  | 王 | 计算机 |
| S6  | 徐 | 计算机 |
+-----+----+--------+
2 rows in set (0.00 sec)

mysql> SELECT SNO,SN,DEPT FROM S WHERE DEPT=(SELECT DEPT FROM S WHERE
SN='王');
+-----+----+--------+
| SNO | SN | DEPT   |
+-----+----+--------+
| S2  | 王 | 计算机 |
| S6  | 徐 | 计算机 |
+-----+----+--------+
2 rows in set (0.00 sec)
```

LIMIT 语句

```
# 取前n条数据
LIMIT n
# 取第n行的数据
LIMIT n-1, 1
# 取第n行到第m行之间的数据
LIMIT n-1,m-n+1
```

```
mysql> SELECT * FROM C;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |    4 |
| C2  | 离散数学 |    2 |
| C3  | 操作系统 |    3 |
| C4  | 数据结构 |    4 |
| C5  | 数据库   |    4 |
| C6  | 汇编语言 |    3 |
| C7  | 信息基础 |    2 |
+-----+----------+------+
7 rows in set (0.00 sec)

mysql> SELECT * FROM C LIMIT 3;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |    4 |
| C2  | 离散数学 |    2 |
| C3  | 操作系统 |    3 |
+-----+----------+------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM C LIMIT 3,1;
+-----+----------+------+
```

```
| CNO | CN       | CT   |
+-----+----------+------+
| C4  | 数据结构 |    4 |
+-----+----------+------+
1 row in set (0.00 sec)
```

储存查询结果到表中

```
# MySQL 语法
CREATE TABLE table_name
AS
SELECT ...;
```

```
mysql> CREATE TABLE C_1TO3 AS SELECT * FROM C LIMIT 3;
Query OK, 3 rows affected (0.75 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> show tables;
+-------------------+
| Tables_in_mysql_0 |
+-------------------+
| c                 |
| c_1to3            |
| deptage           |
| s                 |
| sc                |
+-------------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM C_1TO3;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |    4 |
| C2  | 离散数学 |    2 |
| C3  | 操作系统 |    3 |
+-----+----------+------+
3 rows in set (0.00 sec)
```

数据更新

INSERT 、 UPDATE 、 DELETE

插入数据

```
# 语法
INSERT INTO <table_name>[column_name1, column_name2,...]
VALUES
(value1, value2, ...);
```

注意:

1. 若没有指定任何列，则默认修改全部列。
2. 若某些列名没有出现在into子句中，则在新纪录中这些列的值取空值。

表 S

表 C

表 SC

- 插入单个元组

```
mysql> INSERT INTO C
    -> VALUES
    -> ('C1','C语言',4);
Query OK, 1 row affected (0.07 sec)
```

- 插入多个元组

```
mysql> INSERT INTO C
    -> VALUES
    -> ('C2','离散数学',2),
    -> ('C3','操作系统',3),
    -> ('C4','数据结构',4),
    -> ('C5','数据库',4),
    -> ('C6','汇编语言',3),
    -> ('C7','信息基础',2);
Query OK, 6 rows affected (0.09 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

查看表的全部内容

```
SELECT * FROM <table_name>;
```

```
mysql> SELECT * FROM C;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |    4 |
| C2  | 离散数学 |    2 |
| C3  | 操作系统 |    3 |
| C4  | 数据结构 |    4 |
| C5  | 数据库   |    4 |
| C6  | 汇编语言 |    3 |
| C7  | 信息基础 |    2 |
+-----+----------+------+
7 rows in set (0.00 sec)
```

- 插入子查询结果

```
mysql> CREATE TABLE DEPTAGE
    -> (
    -> DEPT varchar(20),
    -> AVGAGE tinyint
    -> ) default charset utf8;
Query OK, 0 rows affected, 1 warning (0.35 sec)

mysql> INSERT INTO DEPTAGE(DEPT,AVGAGE)
    -> SELECT DEPT,AVG(AGE) FROM S GROUP BY DEPT;
```

```
Query OK, 3 rows affected (0.13 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM DEPTAGE;
+--------+--------+
| DEPT   | AVGAGE |
+--------+--------+
| 信息   |     20 |
| 计算机 |     20 |
| 自动化 |     21 |
+--------+--------+
3 rows in set (0.00 sec)
```

- 插入某些列

```
mysql> INSERT INTO SC(SNO,CNO)
    -> VALUES
    -> ('S7','C1');
Query OK, 1 row affected (0.15 sec)

mysql> SELECT * FROM SC;
+-----+-----+-------+
| SNO | CNO | SCORE |
+-----+-----+-------+
| S1  | C1  |    90 |
| S1  | C2  |    85 |
| S2  | C1  |    84 |
| S2  | C2  |    94 |
| S2  | C3  |    83 |
| S3  | C1  |    73 |
| S3  | C4  |    88 |
| S3  | C5  |    85 |
| S3  | C7  |    68 |
| S4  | C2  |    65 |
| S4  | C5  |    90 |
| S4  | C6  |    79 |
| S5  | C2  |    89 |
| S7  | C1  |  NULL |
+-----+-----+-------+
14 rows in set (0.00 sec)
```

修改数据

```
# 语法
UPDATE <表名>
SET <列名>=<表达式>[,<列名>=<表达式>]...
[WHERE<条件>]
```

功能：修改指定表中满足WHERE子句条件的元组。

其中SET子句用于指定修改方法，即用<表达式>的值取代相应的属性列的值。如果省略WHERE子句，则表示要修改表中的所有元组。

- 修改某一个元组的值

```
mysql> SELECT * FROM S;
```

```
+-----+------+-----+-----+--------+
| SNO | SN   | SEX | AGE | DEPT   |
+-----+------+-----+-----+--------+
| S1  | 李   | 男  |  20 | 信息   |
| S2  | 王   | 女  |  19 | 计算机 |
| S3  | 陈   | 女  |  23 | 自动化 |
| S4  | 张   | 男  |  18 | 自动化 |
| S5  | 吴   | 女  |  20 | 信息   |
| S6  | 徐   | 女  |  21 | 计算机 |
| S7  | 陈东 | 男  |  19 | 信息   |
+-----+------+-----+-----+--------+
7 rows in set (0.00 sec)

mysql> UPDATE S
    -> SET AGE=22
    -> WHERE SNO='S3';
Query OK, 1 row affected (0.13 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM S;
+-----+------+-----+-----+--------+
| SNO | SN   | SEX | AGE | DEPT   |
+-----+------+-----+-----+--------+
| S1  | 李   | 男  |  20 | 信息   |
| S2  | 王   | 女  |  19 | 计算机 |
| S3  | 陈   | 女  |  22 | 自动化 |
| S4  | 张   | 男  |  18 | 自动化 |
| S5  | 吴   | 女  |  20 | 信息   |
| S6  | 徐   | 女  |  21 | 计算机 |
| S7  | 陈东 | 男  |  19 | 信息   |
+-----+------+-----+-----+--------+
7 rows in set (0.05 sec)
```

- 修改多个元组的值

```
mysql> UPDATE S
    -> SET AGE=AGE+1;
Query OK, 7 rows affected (0.14 sec)
Rows matched: 7  Changed: 7  Warnings: 0

mysql> SELECT * FROM S;
+-----+------+-----+-----+--------+
| SNO | SN   | SEX | AGE | DEPT   |
+-----+------+-----+-----+--------+
| S1  | 李   | 男  |  21 | 信息   |
| S2  | 王   | 女  |  20 | 计算机 |
| S3  | 陈   | 女  |  23 | 自动化 |
| S4  | 张   | 男  |  19 | 自动化 |
| S5  | 吴   | 女  |  21 | 信息   |
| S6  | 徐   | 女  |  22 | 计算机 |
| S7  | 陈东 | 男  |  20 | 信息   |
+-----+------+-----+-----+--------+
7 rows in set (0.00 sec)
```

- 带子查询的修改语句

```
mysql> SELECT * FROM SC;
+-----+-----+-------+
| SNO | CNO | SCORE |
+-----+-----+-------+
| S1  | C1  |    90 |
| S1  | C2  |    85 |
| S2  | C1  |    84 |
| S2  | C2  |    94 |
| S2  | C3  |    83 |
| S3  | C1  |    73 |
| S3  | C4  |    88 |
| S3  | C5  |    85 |
| S3  | C7  |    68 |
| S4  | C2  |    65 |
| S4  | C5  |    90 |
| S4  | C6  |    79 |
| S5  | C2  |    89 |
| S7  | C1  |  NULL |
+-----+-----+-------+
14 rows in set (0.00 sec)

mysql> UPDATE SC
    -> SET SCORE=0
    -> WHERE SNO IN(SELECT SNO FROM S WHERE DEPT='计算机');
Query OK, 3 rows affected (0.15 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> SELECT * FROM SC;
+-----+-----+-------+
| SNO | CNO | SCORE |
+-----+-----+-------+
| S1  | C1  |    90 |
| S1  | C2  |    85 |
| S2  | C1  |     0 |
| S2  | C2  |     0 |
| S2  | C3  |     0 |
| S3  | C1  |    73 |
| S3  | C4  |    88 |
| S3  | C5  |    85 |
| S3  | C7  |    68 |
| S4  | C2  |    65 |
| S4  | C5  |    90 |
| S4  | C6  |    79 |
| S5  | C2  |    89 |
| S7  | C1  |  NULL |
+-----+-----+-------+
14 rows in set (0.00 sec)
```

删除数据

```
# 语法
DELETE [FROM] <table_name>[WHERE <条件>]
```

功能：从指定表中删除满足WHERE子句条件的所有元组。

如果省略WHERE子句，则表示删除表中的全部元组，但表的定义仍在字典中。还有一种方法可以快速删除表中全部元组，语法为

```
TRUNCATE <table_name>
```

。

- 删除某一个元组的值

```
mysql> DELETE FROM S
    -> WHERE SNO='S7';
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`mysql_0`.`sc`, CONSTRAINT `fk_SNO_SC` FOREIGN KEY (`SNO`) REFERENCES `s` (`SNO`))
    # 删除失败，因为S表有一个外部键与之关联
mysql> DELETE FROM SC
    -> WHERE SNO='S7';
Query OK, 1 row affected (0.08 sec)

mysql> SELECT * FROM SC;
+-----+-----+-------+
| SNO | CNO | SCORE |
+-----+-----+-------+
| S1  | C1  |    90 |
| S1  | C2  |    85 |
| S2  | C1  |     0 |
| S2  | C2  |     0 |
| S2  | C3  |     0 |
| S3  | C1  |    73 |
| S3  | C4  |    88 |
| S3  | C5  |    85 |
| S3  | C7  |    68 |
| S4  | C2  |    65 |
| S4  | C5  |    90 |
| S4  | C6  |    79 |
| S5  | C2  |    89 |
+-----+-----+-------+
13 rows in set (0.00 sec)
```

- 删除多个元组的值

```
mysql> DELETE FROM SC; # 删除所有记录
Query OK, 13 rows affected (0.17 sec)

mysql> SELECT * FROM SC;
Empty set (0.00 sec)

mysql> INSERT INTO SC
    -> VALUES
    -> ('S1','C1',90),
    -> ('S1','C2',85),
    -> ('S2','C1',84),
    -> ('S2','C2',94);
Query OK, 4 rows affected (0.15 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> TRUNCATE SC; # 删除所有记录
Query OK, 0 rows affected (0.95 sec)

mysql> SELECT * FROM SC;
Empty set (0.11 sec)
```

- 带子查询的删除语句

```
mysql> INSERT INTO SC
    -> VALUES
    -> ('S1','C1',90),
    -> ('S1','C2',85),
    -> ('S2','C1',84),
    -> ('S2','C2',94),
    -> ('S2','C3',83),
    -> ('S3','C1',73),
    -> ('S3','C7',68),
    -> ('S3','C4',88),
    -> ('S3','C5',85),
    -> ('S4','C2',65),
    -> ('S4','C5',90),
    -> ('S4','C6',79),
    -> ('S5','C2',89);
Query OK, 13 rows affected (0.12 sec)
Records: 13  Duplicates: 0  Warnings: 0

mysql> DELETE FROM SC
    -> WHERE SNO IN(SELECT SNO FROM S WHERE DEPT='计算机');
Query OK, 3 rows affected (0.13 sec)

mysql> SELECT * FROM SC;
+-----+-----+-------+
| SNO | CNO | SCORE |
+-----+-----+-------+
| S1  | C1  |    90 |
| S1  | C2  |    85 |
| S3  | C1  |    73 |
| S3  | C4  |    88 |
| S3  | C5  |    85 |
| S3  | C7  |    68 |
| S4  | C2  |    65 |
| S4  | C5  |    90 |
| S4  | C6  |    79 |
| S5  | C2  |    89 |
+-----+-----+-------+
10 rows in set (0.01 sec)
```

视图

视图是从一个或几个基本表(视图)导出的表，它与基本表不同，是一个虚表。

创建视图

```
# 语法
CREATE VIEW <view_name>[(column_name1,column_name2, ...)]
AS <子查询>
```

注意：

- 子查询可以是任意复杂的SELECT语句，但是不允许有ORDER BY子句和DISTINCT短语。
- 组成视图的属性列名要么全部省略，要么全部指定

```
mysql> show tables;
+-------------------+
| Tables_in_mysql_0 |
+-------------------+
| c                 |
| c_1to3            |
| deptage           |
| s                 |
| sc                |
+-------------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM C;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |    4 |
| C2  | 离散数学 |    2 |
| C3  | 操作系统 |    3 |
| C4  | 数据结构 |    4 |
| C5  | 数据库   |    4 |
| C6  | 汇编语言 |    3 |
| C7  | 信息基础 |    2 |
+-----+----------+------+
7 rows in set (0.00 sec)

mysql> CREATE VIEW IS_C
    -> AS SELECT * FROM C WHERE CT>=3;
Query OK, 0 rows affected (0.12 sec)

mysql> show tables;
+-------------------+
| Tables_in_mysql_0 |
+-------------------+
| c                 |
| c_1to3            |
| deptage           |
| is_c              |
| s                 |
| sc                |
+-------------------+
6 rows in set (0.00 sec)
```

删除视图

```
# 语法
DROP VIEW <view_name>
```

```
mysql> DROP VIEW IS_C;
Query OK, 0 rows affected (0.16 sec)

mysql> show tables;
+-------------------+
| Tables_in_mysql_0 |
+-------------------+
```

```
| c                 |
| c_1to3            |
| deptage           |
| s                 |
| sc                |
+-------------------+
5 rows in set (0.00 sec)
```

查询视图

```
# 与基本表的查询语法一致
mysql> SELECT * FROM IS_C;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |    4 |
| C3  | 操作系统 |    3 |
| C4  | 数据结构 |    4 |
| C5  | 数据库   |    4 |
| C6  | 汇编语言 |    3 |
+-----+----------+------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM IS_C WHERE CT>3;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |    4 |
| C4  | 数据结构 |    4 |
| C5  | 数据库   |    4 |
+-----+----------+------+
3 rows in set (0.00 sec)
```

更新视图

与基本表的更新的语法一致。有插入(INSERT)、删除(DELETE)、修改(UPDATE)三类操作。但是并不是所
有的视图都可以更新的。

```
mysql> UPDATE IS_C SET CT=35 WHERE CNO='C1';
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM IS_C;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |   35 |
| C3  | 操作系统 |    3 |
| C4  | 数据结构 |    4 |
| C5  | 数据库   |    4 |
| C6  | 汇编语言 |    3 |
+-----+----------+------+
5 rows in set (0.00 sec)

```

```
mysql> SELECT * FROM C;
+-----+----------+------+
| CNO | CN       | CT   |
+-----+----------+------+
| C1  | C语言    |   35 |
| C2  | 离散数学 |    2 |
| C3  | 操作系统 |    3 |
| C4  | 数据结构 |    4 |
| C5  | 数据库   |    4 |
| C6  | 汇编语言 |    3 |
| C7  | 信息基础 |    2 |
+-----+----------+------+
7 rows in set (0.00 sec)
```

数据备份与恢复

备份

```
-- 语法：
# mysqldump基本语法：
    # 1.备份一个数据库
mysqldump -u username -p database_name table_name1 table_name2 ... > BackupName.sql
    # 2.备份多个数据库
mysqldump -u username -p --databases dbname1,dbname2,... > BackupName.sql
    # 3.备份所有数据库
mysqldump -u username -p -all-databases > BackupName.sql
# 注意：这些命令与上方的命令不同，是在终端或命令行中执行的，而不是在mysql的shell中，要退出mysql。恢复也是。
```

```
C:\Users\Administrator>mysqldump -u root -p mysql_0 > E:/mysql_0.sql
Enter password: ******
```

其中：

- 需要备份的表为空则整个数据库备份；
- BackupName.sql参数表设计备份文件的名称，文件名前面可以加上一个绝对路径。通常将数据库保存成一个 sql 文件；

恢复

```
# 语法
mysql -u username -p database_name < backup.sql
```

backup.sql 表示已经备份好的数据库的路径

```
C:\Users\Administrator>mysql -u root -p mysql_0 < E:/mysql_0.sql
Enter password: ******
```