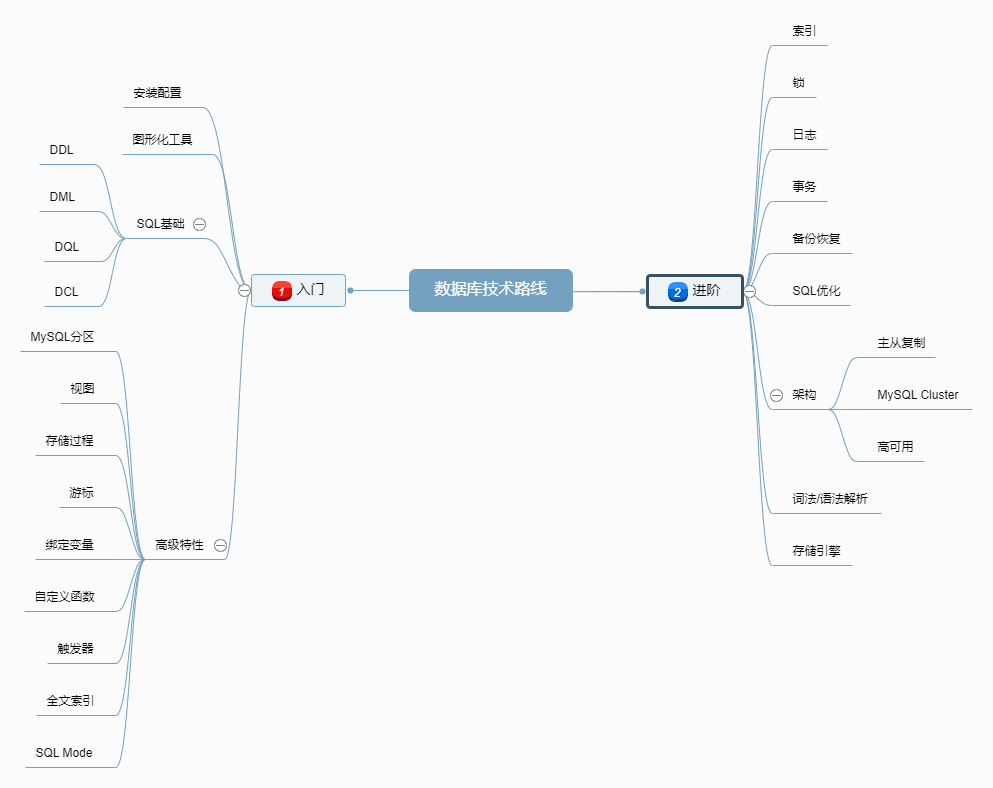
# 思维导图



# 入门

数据库入门首选要学会基本环境搭建，然后选择一款顺手的图形界面工具，学习各种SQL语句，包括基本建库建表的DDL语句，然后增删改查语句，再掌握了基本的操作之后，需要学会使用一些高级特性，比如视图、存储过程、游标、自定义函数等。

## 安装配置

1. Windows和Linux系统MySQL安装
2. MySQL基本配置
3. 启动和关闭MySQL

## 图形化工具

### MySQL Workbench

### phpMyAdmin

## SQL基础

SQL语句变化多端，但是真正实际开发中使用的无外乎

### DDL

#### 数据类型

1. 整值类型：

整数类型：TINYINT、SMALLINT、MEDIUMINT、INT、INTEGER、BIGINT

浮点数类型：FLOAT、DOUBLE

定点数类型：DEC、DECIMAL

位类型：BIT

1. 字符串类型：

CHAR/VARCHAR

BINARY、VARBINARY

TEXT、BLOB

ENUM、SET

1. 日期类型选择

DATE、DATETIME、TIMESTAMP、TIME、YEAR

#### 字符集

1. 概述
2. Unicode
3. 汉字编码
4. 字符集选择
5. MySQL字符集

#### 表类型选择

1. 存储引擎概述
2. 各种存储引擎基本特性
3. 如何选择存储引擎

#### 约束

1. 主键/外键
2. 索引/唯一索引
3. not null

#### CREATE

##### 库

建库：

CREATE DATABASE [IF NOT EXISTS] <数据库名>

[[DEFAULT] CHARACTER SET <字符集名>]

[[DEFAULT] COLLATE <校对规则名>];

[ ]中的内容是可选的。语法说明如下：

<数据库名>：创建数据库的名称。MySQL 的数据存储区将以目录方式表示 MySQL 数据库，因此数据库名称必须符合操作系统的文件夹命名规则，不能以数字开头，尽量要有实际意义。注意在 MySQL 中不区分大小写。

IF NOT EXISTS：在创建数据库之前进行判断，只有该数据库目前尚不存在时才能执行操作。此选项可以用来避免数据库已经存在而重复创建的错误。

[DEFAULT] CHARACTER SET：指定数据库的字符集。指定字符集的目的是为了避免在数据库中存储的数据出现乱码的情况。如果在创建数据库时不指定字符集，那么就使用系统的默认字符集。

[DEFAULT] COLLATE：指定字符集的默认校对规则。

MySQL 的字符集（CHARACTER）和校对规则（COLLATION）是两个不同的概念。字符集是用来定义 MySQL 存储字符串的方式，校对规则定义了比较字符串的方式。后面我们会单独讲解 MySQL 的字符集和校对规则。

##### 表

建表：

CREATE TABLE <表名> ([表定义选项])[表选项][分区选项];

其中，[表定义选项]的格式为：

<列名1> <类型1> [,…] <列名n> <类型n>

CREATE TABLE 命令语法比较多，其主要是由表创建定义（create-definition）、表选项（table-options）和分区选项（partition-options）所组成的。  
 这里首先描述一个简单的新建表的例子，然后重点介绍 CREATE TABLE 命令中的一些主要的语法知识点。  
 CREATE TABLE 语句的主要语法及使用说明如下：

CREATE TABLE：用于创建给定名称的表，必须拥有表CREATE的权限。

<表名>：指定要创建表的名称，在 CREATE TABLE 之后给出，必须符合标识符命名规则。表名称被指定为 db\_name.tbl\_name，以便在特定的数据库中创建表。无论是否有当前数据库，都可以通过这种方式创建。在当前数据库中创建表时，可以省略 db-name。如果使用加引号的识别名，则应对数据库和表名称分别加引号。例如，'mydb'.'mytbl' 是合法的，但 'mydb.mytbl' 不合法。

<表定义选项>：表创建定义，由列名（col\_name）、列的定义（column\_definition）以及可能的空值说明、完整性约束或表索引组成。

默认的情况是，表被创建到当前的数据库中。若表已存在、没有当前数据库或者数据库不存在，则会出现错误。

提示：使用 CREATE TABLE 创建表时，必须指定以下信息：

要创建的表的名称不区分大小写，不能使用SQL语言中的关键字，如DROP、ALTER、INSERT等。

数据表中每个列（字段）的名称和数据类型，如果创建多个列，要用逗号隔开。

#### DROP

##### 库

在 [MySQL](http://c.biancheng.net/mysql/" \t "http://c.biancheng.net/view/_blank) 中，当需要删除已创建的数据库时，可以使用 DROP DATABASE 语句。其语法格式为：

DROP DATABASE [ IF EXISTS ] <数据库名>

语法说明如下：

<数据库名>：指定要删除的数据库名。

IF EXISTS：用于防止当数据库不存在时发生错误。

DROP DATABASE：删除数据库中的所有表格并同时删除数据库。使用此语句时要非常小心，以免错误删除。如果要使用 DROP DATABASE，需要获得数据库 DROP 权限。

注意：MySQL 安装后，系统会自动创建名为 information\_schema 和 mysql 的两个系统数据库，系统数据库存放一些和数据库相关的信息，如果删除了这两个数据库，MySQL 将不能正常工作。

##### 表

使用 DROP TABLE 语句可以删除一个或多个数据表，语法格式如下：

DROP TABLE [IF EXISTS] 表名1 [ ,表名2, 表名3 ...]

对语法格式的说明如下：

表名1, 表名2, 表名3 ...表示要被删除的数据表的名称。DROP TABLE 可以同时删除多个表，只要将表名依次写在后面，相互之间用逗号隔开即可。

IF EXISTS 用于在删除数据表之前判断该表是否存在。如果不加 IF EXISTS，当数据表不存在时 MySQL 将提示错误，中断 SQL 语句的执行；加上 IF EXISTS 后，当数据表不存在时 SQL 语句可以顺利执行，但是会发出警告（warning）。

两点注意：

用户必须拥有执行 DROP TABLE 命令的权限，否则数据表不会被删除。

表被删除时，用户在该表上的权限不会自动删除。

##### 字段

#### ALTER

##### 库

修改库：

ALTER DATABASE [数据库名] {

[ DEFAULT ] CHARACTER SET <字符集名> |

[ DEFAULT ] COLLATE <校对规则名>}

语法说明如下：

ALTER DATABASE 用于更改数据库的全局特性。

使用 ALTER DATABASE 需要获得数据库 ALTER 权限。

数据库名称可以忽略，此时语句对应于默认数据库。

CHARACTER SET 子句用于更改默认的数据库字符集。

##### 表

修改表：

ALTER TABLE <表名> [修改选项]

修改选项的语法格式如下：

{ ADD COLUMN <列名> <类型>  
 | CHANGE COLUMN <旧列名> <新列名> <新列类型>  
 | ALTER COLUMN <列名> { SET DEFAULT <默认值> | DROP DEFAULT }  
 | MODIFY COLUMN <列名> <类型>  
 | DROP COLUMN <列名>  
 | RENAME TO <新表名>  
 | CHARACTER SET <字符集名>  
 | COLLATE <校对规则名> }

MySQL 通过 ALTER TABLE 语句来实现表名的修改，语法规则如下：

ALTER TABLE <旧表名> RENAME [TO] <新表名>；

其中，TO 为可选参数，使用与否均不影响结果。

MySQL 通过 ALTER TABLE 语句来实现表字符集的修改，语法规则如下：

ALTER TABLE 表名 [DEFAULT] CHARACTER SET <字符集名> [DEFAULT] COLLATE <校对规则名>;

其中，DEFAULT 为可选参数，使用与否均不影响结果。

##### 字段

MySQL 中修改表字段名的语法规则如下：

ALTER TABLE <表名> CHANGE <旧字段名> <新字段名> <新数据类型>；

其中：

旧字段名：指修改前的字段名；

新字段名：指修改后的字段名；

新数据类型：指修改后的数据类型，如果不需要修改字段的数据类型，可以将新数据类型设置成与原来一样，但数据类型不能为空。

修改字段的数据类型就是把字段的数据类型转换成另一种数据类型。在 MySQL 中修改字段数据类型的语法规则如下：

ALTER TABLE <表名> MODIFY <字段名> <数据类型>

其中：

表名：指要修改数据类型的字段所在表的名称；

字段名：指需要修改的字段；

数据类型：指修改后字段的新数据类型。

删除字段是将数据表中的某个字段从表中移除，语法格式如下：

ALTER TABLE <表名> DROP <字段名>；

其中，“字段名”指需要从表中删除的字段的名称。

### DML

#### INSERT

INSERT 语句有两种语法形式，分别是 INSERT…VALUES 语句和 INSERT…SET 语句。

1) INSERT…VALUES语句

INSERT VALUES 的语法格式为：

INSERT INTO <表名> [ <列名1> [ , … <列名n>] ]  
 VALUES (值1) [… , (值n) ];

语法说明如下。

<表名>：指定被操作的表名。

<列名>：指定需要插入数据的列名。若向表中的所有列插入数据，则全部的列名均可以省略，直接采用 INSERT<表名>VALUES(…) 即可。

VALUES 或 VALUE 子句：该子句包含要插入的数据清单。数据清单中数据的顺序要和列的顺序相对应。

2) INSERT…SET语句

语法格式为：

INSERT INTO <表名>  
 SET <列名1> = <值1>,  
        <列名2> = <值2>,  
        …

此语句用于直接给表中的某些列指定对应的列值，即要插入的数据的列名在 SET 子句中指定，col\_name 为指定的列名，等号后面为指定的数据，而对于未指定的列，列值会指定为该列的默认值。  
  
 由 INSERT 语句的两种形式可以看出：

使用 INSERT…VALUES 语句可以向表中插入一行数据，也可以插入多行数据；

使用 INSERT…SET 语句可以指定插入行中每列的值，也可以指定部分列的值；

INSERT…SELECT 语句向表中插入其他表的数据。

采用 INSERT…SET 语句可以向表中插入部分列的值，这种方式更为灵活；

INSERT…VALUES 语句可以一次插入多条数据。

在 MySQL 中，用单条 INSERT 语句处理多个插入要比使用多条 INSERT 语句更快。  
 当使用单条 INSERT 语句插入多行数据的时候，只需要将每行数据用圆括号括起来即可。

#### UPDATE

使用 UPDATE 语句修改单个表，语法格式为：

UPDATE <表名> SET 字段 1=值 1 [,字段 2=值 2… ] [WHERE 子句 ]  
[ORDER BY 子句] [LIMIT 子句]

语法说明如下：

<表名>：用于指定要更新的表名称。

SET 子句：用于指定表中要修改的列名及其列值。其中，每个指定的列值可以是表达式，也可以是该列对应的默认值。如果指定的是默认值，可用关键字 DEFAULT 表示列值。

WHERE 子句：可选项。用于限定表中要修改的行。若不指定，则修改表中所有的行。

ORDER BY 子句：可选项。用于限定表中的行被修改的次序。

LIMIT 子句：可选项。用于限定被修改的行数。

注意：修改一行数据的多个列值时，SET 子句的每个值用逗号分开即可。

#### DELETE

使用 DELETE 语句从单个表中删除数据，语法格式为：

DELETE FROM <表名> [WHERE 子句] [ORDER BY 子句] [LIMIT 子句]

语法说明如下：

<表名>：指定要删除数据的表名。

ORDER BY 子句：可选项。表示删除时，表中各行将按照子句中指定的顺序进行删除。

WHERE 子句：可选项。表示为删除操作限定删除条件，若省略该子句，则代表删除该表中的所有行。

LIMIT 子句：可选项。用于告知服务器在控制命令被返回到客户端前被删除行的最大值。

注意：在不使用 WHERE 条件的时候，将删除所有数据。

#### TRUNCATE

TRUNCATE 关键字用于完全清空一个表。其语法格式如下：

TRUNCATE [TABLE] 表名

其中，TABLE 关键字可省略。

### DQL

#### 基本查询

查询语句：

SELECT

{\* | <字段列名>}

[

FROM <表 1>, <表 2>…

[WHERE <表达式>

[GROUP BY <group by definition>

[HAVING <expression> [{<operator> <expression>}…]]

[ORDER BY <order by definition>]

[LIMIT[<offset>,] <row count>]

]

其中，各条子句的含义如下：

{\*|<字段列名>}包含星号通配符的字段列表，表示所要查询字段的名称。

<表 1>，<表 2>…，表 1 和表 2 表示查询数据的来源，可以是单个或多个。

WHERE <表达式>是可选项，如果选择该项，将限定查询数据必须满足该查询条件。

GROUP BY< 字段 >，该子句告诉 MySQL 如何显示查询出来的数据，并按照指定的字段分组。

[ORDER BY< 字段 >]，该子句告诉 MySQL 按什么样的顺序显示查询出来的数据，可以进行的排序有升序（ASC）和降序（DESC），默认情况下是升序。

[LIMIT[<offset>，]<row count>]，该子句告诉 MySQL 每次显示查询出来的数据条数。

##### 别名

语法：

<表名> [AS] <别名>

其中各子句的含义如下：

<表名>：数据库中存储的数据表的名称。

<别名>：查询时指定的表的新名称。

AS关键字可以省略，省略后需要将表名和别名用空格隔开。

注意：表的别名不能与该数据库的其它表同名。字段的别名不能与该表的其它字段同名。在条件表达式中不能使用字段的别名，否则会出现“ERROR 1054 (42S22): Unknown column”这样的错误提示信息。

##### 条件查询

语法：

WHERE 查询条件

查询条件可以是：

带比较运算符和逻辑运算符的查询条件

带 BETWEEN AND 关键字的查询条件

带 IS NULL 关键字的查询条件

带 IN 关键字的查询条件

带 LIKE 关键字的查询条件

###### 模糊查询

在 MySQL 中，LIKE 关键字主要用于搜索匹配字段中的指定内容。其语法格式如下：

[NOT] LIKE  '字符串'

其中：

NOT ：可选参数，字段中的内容与指定的字符串不匹配时满足条件。

字符串：指定用来匹配的字符串。“字符串”可以是一个很完整的字符串，也可以包含通配符。

LIKE 关键字支持百分号“%”和下划线“\_”通配符。

通配符是一种特殊语句，主要用来模糊查询。当不知道真正字符或者懒得输入完整名称时，可以使用通配符来代替一个或多个真正的字符。

###### 范围查找

使用 BETWEEN AND 的基本语法格式如下：

[NOT] BETWEEN 取值1 AND 取值2

其中：

NOT：可选参数，表示指定范围之外的值。如果字段值不满足指定范围内的值，则这些记录被返回。

取值1：表示范围的起始值。

取值2：表示范围的终止值。

###### 空值查询

MySQL 提供了 IS NULL 关键字，用来判断字段的值是否为空值（NULL）。空值不同于 0，也不同于空字符串。  
 如果字段的值是空值，则满足查询条件，该记录将被查询出来。如果字段的值不是空值，则不满足查询条件。  
  
 使用 IS NULL 的基本语法格式如下：

IS [NOT] NULL

其中，“NOT”是可选参数，表示字段值不是空值时满足条件。

##### 分组/聚合

语法：

GROUP BY  <字段名>

##### 过滤分组

语法：

HAVING <查询条件>

HAVING 关键字和 WHERE 关键字都可以用来过滤数据，且 HAVING 支持 WHERE 关键字中所有的操作符和语法。  
  
 但是 WHERE 和 HAVING 关键字也存在以下几点差异：

一般情况下，WHERE 用于过滤数据行，而 HAVING 用于过滤分组。

WHERE 查询条件中不可以使用聚合函数，而 HAVING 查询条件中可以使用聚合函数。

WHERE 在数据分组前进行过滤，而 HAVING 在数据分组后进行过滤 。

WHERE 针对数据库文件进行过滤，而 HAVING 针对查询结果进行过滤。也就是说，WHERE 根据数据表中的字段直接进行过滤，而 HAVING 是根据前面已经查询出的字段进行过滤。

WHERE 查询条件中不可以使用字段别名，而 HAVING 查询条件中可以使用字段别名。

##### 排序

语法：

ORDER BY <字段名> [ASC|DESC]

语法说明如下。

字段名：表示需要排序的字段名称，多个字段时用逗号隔开。

ASC|DESC：ASC表示字段按升序排序；DESC表示字段按降序排序。其中ASC为默认值。

使用 ORDER BY 关键字应该注意以下几个方面：

ORDER BY 关键字后可以跟子查询（关于子查询后面教程会详细讲解，这里了解即可）。

当排序的字段中存在空值时，ORDER BY 会将该空值作为最小值来对待。

ORDER BY 指定多个字段进行排序时，MySQL 会按照字段的顺序从左到右依次进行排序。

##### 去重

语法：

SELECT DISTINCT <字段名> FROM <表名>;

其中，“字段名”为需要消除重复记录的字段名称，多个字段时用逗号隔开。  
  
 使用DISTINCT关键字时需要注意以下几点：

DISTINCT 关键字只能在 SELECT 语句中使用。

在对一个或多个字段去重时，DISTINCT 关键字必须在所有字段的最前面。

如果 DISTINCT 关键字后有多个字段，则会对多个字段进行组合去重，也就是说，只有多个字段组合起来完全是一样的情况下才会被去重。

##### 分页

语法：

LIMIT 初始位置，记录数

其中，“初始位置”表示从哪条记录开始显示；“记录数”表示显示记录的条数。第一条记录的位置是 0，第二条记录的位置是 1。后面的记录依次类推。  
  
 注意：LIMIT 后的两个参数必须都是正整数。

##### 表连接

###### 内连接

内连接（INNER JOIN）主要通过设置连接条件的方式，来移除查询结果中某些数据行的交叉连接。简单来说，就是利用条件表达式来消除交叉连接的某些数据行。  
 内连接使用 INNER JOIN 关键字连接两张表，并使用 ON 子句来设置连接条件。如果没有连接条件，INNER JOIN 和 CROSS JOIN 在语法上是等同的，两者可以互换。  
  
 内连接的语法格式如下：

SELECT <字段名> FROM <表1> INNER JOIN <表2> [ON子句]

语法说明如下。

字段名：需要查询的字段名称。

<表1><表2>：需要内连接的表名。

INNER JOIN ：内连接中可以省略 INNER 关键字，只用关键字 JOIN。

ON 子句：用来设置内连接的连接条件。

INNER JOIN 也可以使用 WHERE 子句指定连接条件，但是 INNER JOIN ... ON 语法是官方的标准写法，而且 WHERE 子句在某些时候会影响查询的性能。

多个表内连接时，在 FROM 后连续使用 INNER JOIN 或 JOIN 即可。

###### 外连接

左外连接

左外连接又称为左连接，使用 LEFT OUTER JOIN 关键字连接两个表，并使用 ON 子句来设置连接条件。  
  
 左连接的语法格式如下：

SELECT <字段名> FROM <表1> LEFT OUTER JOIN <表2> <ON子句>

语法说明如下。

字段名：需要查询的字段名称。

<表1><表2>：需要左连接的表名。

LEFT OUTER JOIN：左连接中可以省略 OUTER 关键字，只使用关键字 LEFT JOIN。

ON 子句：用来设置左连接的连接条件，不能省略。

上述语法中，“表1”为基表，“表2”为参考表。左连接查询时，可以查询出“表1”中的所有记录和“表2”中匹配连接条件的记录。如果“表1”的某行在“表2”中没有匹配行，那么在返回结果中，“表2”的字段值均为空值（NULL）。

右外连接

右外连接又称为右连接，右连接是左连接的反向连接。使用 RIGHT OUTER JOIN 关键字连接两个表，并使用 ON 子句来设置连接条件。  
  
 右连接的语法格式如下：

SELECT <字段名> FROM <表1> RIGHT OUTER JOIN <表2> <ON子句>

语法说明如下。

字段名：需要查询的字段名称。

<表1><表2>：需要右连接的表名。

RIGHT OUTER JOIN：右连接中可以省略 OUTER 关键字，只使用关键字 RIGHT JOIN。

ON 子句：用来设置右连接的连接条件，不能省略。

与左连接相反，右连接以“表2”为基表，“表1”为参考表。右连接查询时，可以查询出“表2”中的所有记录和“表1”中匹配连接条件的记录。如果“表2”的某行在“表1”中没有匹配行，那么在返回结果中，“表1”的字段值均为空值（NULL）。

###### 交叉连接

交叉连接的语法格式如下：

SELECT <字段名> FROM <表1> CROSS JOIN <表2> [WHERE子句]

或

SELECT <字段名> FROM <表1>, <表2> [WHERE子句]

语法说明如下：

字段名：需要查询的字段名称。

<表1><表2>：需要交叉连接的表名。

WHERE 子句：用来设置交叉连接的查询条件。

注意：多个表交叉连接时，在 FROM 后连续使用 CROSS JOIN 或,即可。以上两种语法的返回结果是相同的，但是第一种语法才是官方建议的标准写法。

##### 并集

##### 子查询

子查询在 WHERE 中的语法格式如下：

WHERE <表达式> <操作符> (子查询)

其中，操作符可以是比较运算符和 IN、NOT IN、EXISTS、NOT EXISTS 等关键字。

1）IN | NOT IN

当表达式与子查询返回的结果集中的某个值相等时，返回 TRUE，否则返回 FALSE；若使用关键字 NOT，则返回值正好相反。

2）EXISTS | NOT EXISTS

用于判断子查询的结果集是否为空，若子查询的结果集不为空，返回 TRUE，否则返回 FALSE；若使用关键字 NOT，则返回的值正好相反。

#### 运算符

##### 算数运算符

加（+）、减（-）、乘（\*）、除（/，DIV）和模运算（%，MOD）

##### 比较运算符

=、<>/!=、<=>、<、<=、>、>=、BETWEEN、IN、IS NULL、IS NOT NULL、LIKE、REGEXP、RLIKE

##### 逻辑运算符

NOT/!、AND/&&、OR/||、XOR

##### 位运算符

&、|、^、~、>>、<<

##### 运算符优先级

#### 常用函数

##### 字符串函数

CONCAT()、INSERT()、LOWER()、UPPER()、LEFT()、RIGHT()、LPAD()、RPAD()、LTRIM()、RTRIM()、REPEAT()、REPLACE()、STRCMP()、TRIM()、SUBSTRING()

##### 数值函数

ABS()、CEIL()、FLOOR()、MOD()、RAND()、ROUND()、TRUNCATE()

##### 日期和时间函数

CURDATE()、CURTIME()、NOW()、UNIX\_TIMESTAMP()、FROM\_UNIXTIME()、WEEK()、YEAR()、HOUR()、MINUTE()、MONTHNAME()、DATE\_FORMAT()、DATE\_ADD()、DATEDIFF()

##### 流程函数

IF、IFNULL、CASE WHEN

##### 其他函数

DATABASE()、VERSION()、USER()、INET\_ATON()、INET\_NTOA()、PASSWORD()、MD5()

### DCL

#### 用户管理

##### 创建用户

可以使用 CREATE USER 语句来创建 MySQL 用户，并设置相应的密码。其基本语法格式如下：

CREATE USER <用户> [ IDENTIFIED BY [ PASSWORD ] 'password' ] [ ,用户 [ IDENTIFIED BY [ PASSWORD ] 'password' ]]

参数说明如下：

1) 用户

指定创建用户账号，格式为 user\_name'@'host\_name。这里的user\_name是用户名，host\_name为主机名，即用户连接 MySQL 时所用主机的名字。如果在创建的过程中，只给出了用户名，而没指定主机名，那么主机名默认为“%”，表示一组主机，即对所有主机开放权限。

3) IDENTIFIED BY子句

用于指定用户密码。新用户可以没有初始密码，若该用户不设密码，可省略此子句。

2) PASSWORD 'password'

PASSWORD 表示使用哈希值设置密码，该参数可选。如果密码是一个普通的字符串，则不需要使用 PASSWORD 关键字。'password' 表示用户登录时使用的密码，需要用单引号括起来。  
  
使用 CREATE USER 语句时应注意以下几点：

CREATE USER 语句可以不指定初始密码。但是从安全的角度来说，不推荐这种做法。

使用 CREATE USER 语句必须拥有 mysql 数据库的 INSERT 权限或全局 CREATE USER 权限。

使用 CREATE USER 语句创建一个用户后，MySQL 会在 mysql 数据库的 user 表中添加一条新记录。

CREATE USER 语句可以同时创建多个用户，多个用户用逗号隔开。

新创建的用户拥有的权限很少，它们只能执行不需要权限的操作。如登录 MySQL、使用 SHOW 语句查询所有存储引擎和字符集的列表等。如果两个用户的用户名相同，但主机名不同，MySQL 会将它们视为两个用户，并允许为这两个用户分配不同的权限集合。

##### 修改用户

在 [MySQL](http://c.biancheng.net/mysql/" \t "http://c.biancheng.net/view/_blank) 中，我们可以使用 RENAME USER 语句修改一个或多个已经存在的用户账号。  
  
 语法格式如下：

RENAME USER <旧用户> TO <新用户>

其中：

<旧用户>：系统中已经存在的 MySQL 用户账号。

<新用户>：新的 MySQL 用户账号。

使用 RENAME USER 语句时应注意以下几点：

RENAME USER 语句用于对原有的 MySQL 用户进行重命名。

若系统中旧账户不存在或者新账户已存在，该语句执行时会出现错误。

使用 RENAME USER 语句，必须拥有 mysql 数据库的 UPDATE 权限或全局 CREATE USER 权限。

##### 删除用户

使用 DROP USER 语句删除用户的语法格式如下：

DROP USER <用户1> [ , <用户2> ]…

其中，用户用来指定需要删除的用户账号。  
  
 使用 DROP USER 语句应注意以下几点：

DROP USER 语句可用于删除一个或多个用户，并撤销其权限。

使用 DROP USER 语句必须拥有 mysql 数据库的 DELETE 权限或全局 CREATE USER 权限。

在 DROP USER 语句的使用中，若没有明确地给出账户的主机名，则该主机名默认为“%”。

注意：用户的删除不会影响他们之前所创建的表、索引或其他数据库对象，因为 MySQL 并不会记录是谁创建了这些对象。

##### 查看用户权限

在 MySQL 中，可以通过查看 mysql.user 表中的数据记录来查看相应的用户权限，也可以使用 SHOW GRANTS 语句查询用户的权限。  
  
mysql 数据库下的 user 表中存储着用户的基本权限，可以使用 SELECT 语句来查看。SELECT 语句的代码如下：

SELECT \* FROM mysql.user;

要执行该语句，必须拥有对 user 表的查询权限。  
  
 注意：新创建的用户只有登录 MySQL 服务器的权限，没有任何其它权限，不能查询 user 表。  
  
 除了使用 SELECT 语句之外，还可以使用 SHOW GRANTS FOR 语句查看权限。其语法格式如下：

SHOW GRANTS FOR 'username'@'hostname';

其中，username 表示用户名，hostname 表示主机名或主机 IP。

#### GRANT

在 MySQL 中，拥有 GRANT 权限的用户才可以执行 GRANT 语句，其语法格式如下：

GRANT priv\_type [(column\_list)] ON database.table

TO user [IDENTIFIED BY [PASSWORD] 'password']

[, user[IDENTIFIED BY [PASSWORD] 'password']] ...

[WITH with\_option [with\_option]...]

其中：

priv\_type 参数表示权限类型；

columns\_list 参数表示权限作用于哪些列上，省略该参数时，表示作用于整个表；

database.table 用于指定权限的级别；

user 参数表示用户账户，由用户名和主机名构成，格式是“'username'@'hostname'”；

IDENTIFIED BY 参数用来为用户设置密码；

password 参数是用户的新密码。

WITH 关键字后面带有一个或多个 with\_option 参数。这个参数有 5 个选项，详细介绍如下：

GRANT OPTION：被授权的用户可以将这些权限赋予给别的用户；

MAX\_QUERIES\_PER\_HOUR count：设置每个小时可以允许执行 count 次查询；

MAX\_UPDATES\_PER\_HOUR count：设置每个小时可以允许执行 count 次更新；

MAX\_CONNECTIONS\_PER\_HOUR count：设置每小时可以建立 count 个连接;

MAX\_USER\_CONNECTIONS count：设置单个用户可以同时具有的 count 个连接。  
 MySQL 中可以授予的权限有如下几组：

列权限，和表中的一个具体列相关。例如，可以使用 UPDATE 语句更新表 students 中 name 列的值的权限。

表权限，和一个具体表中的所有数据相关。例如，可以使用 SELECT 语句查询表 students 的所有数据的权限。

数据库权限，和一个具体的数据库中的所有表相关。例如，可以在已有的数据库 mytest 中创建新表的权限。

用户权限，和 MySQL 中所有的数据库相关。例如，可以删除已有的数据库或者创建一个新的数据库的权限。

对应地，在 GRANT 语句中可用于指定权限级别的值有以下几类格式：

\*：表示当前数据库中的所有表。

\*.\*：表示所有数据库中的所有表。

db\_name.\*：表示某个数据库中的所有表，db\_name 指定数据库名。

db\_name.tbl\_name：表示某个数据库中的某个表或视图，db\_name 指定数据库名，tbl\_name 指定表名或视图名。

db\_name.routine\_name：表示某个数据库中的某个存储过程或函数，routine\_name 指定存储过程名或函数名。

TO 子句：如果权限被授予给一个不存在的用户，MySQL 会自动执行一条 CREATE USER 语句来创建这个用户，但同时必须为该用户设置密码。

#### REVOKE

在 MySQL 中，可以使用 REVOKE 语句删除某个用户的某些权限（此用户不会被删除），在一定程度上可以保证系统的安全性。例如，如果数据库管理员觉得某个用户不应该拥有 DELETE 权限，那么就可以删除 DELETE 权限。  
  
 使用 REVOKE 语句删除权限的语法格式有两种形式，如下所示：

1）第一种

删除用户某些特定的权限，语法格式如下：

REVOKE priv\_type [(column\_list)]...  
ON database.table  
FROM user [, user]...

REVOKE 语句中的参数与 GRANT 语句的参数意思相同。其中：

priv\_type 参数表示权限的类型；

column\_list 参数表示权限作用于哪些列上，没有该参数时作用于整个表上；

user 参数由用户名和主机名构成，格式为“username'@'hostname'”。

2）第二种

删除特定用户的所有权限，语法格式如下：

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...

删除用户权限需要注意以下几点：

REVOKE 语法和 GRANT 语句的语法格式相似，但具有相反的效果。

要使用 REVOKE 语句，必须拥有 MySQL 数据库的全局 CREATE USER 权限或 UPDATE 权限。

#### 密码管理

##### 修改密码

在 MySQL 中，只有 root 用户可以通过更新 MySQL 数据库来更改密码。使用 root 用户登录到 MySQL 服务器后，可以使用 SET 语句来修改普通用户密码。语法格式如下：

SET PASSWORD FOR 'username'@'hostname' = PASSWORD ('newpwd');

其中，username 参数是普通用户的用户名，hostname 参数是普通用户的主机名，newpwd 是要更改的新密码。  
 注意：新密码必须使用 PASSWORD() 函数来加密，如果不使用 PASSWORD() 加密，也会执行成功，但是用户会无法登录。  
 如果是普通用户修改密码，可省略 FOR 子句来更改自己的密码。语法格式如下：

SET PASSWORD = PASSWORD('newpwd');

root 用户可以使用 mysqladmin 命令来修改密码，mysqladmin 的语法格式如下：

mysqladmin -u username -h hostname -p password "newpwd"

语法参数说明如下：

usermame 指需要修改密码的用户名称，在这里指定为 root 用户；

hostname 指需要修改密码的用户主机名，该参数可以不写，默认是 localhost；

password 为关键字，而不是指旧密码；

newpwd 为新设置的密码，必须用双引号括起来。如果使用单引号会引发错误，可能会造成修改后的密码不是你想要的。

执行完上面的语句，root 用户的密码将被修改为“newpwd”。

##### 重置密码

## 高级特性

### MySQL分区

1. 什么是分区？
2. 分区类型：Range、List、Columns、Hash、Key、子分区
3. 分区管理
4. 分区与性能

### 视图

1. 什么是视图？
2. 视图基本操作：

#### 创建视图

语法格式如下：

CREATE VIEW <视图名> AS <SELECT语句>

语法说明如下。

<视图名>：指定视图的名称。该名称在数据库中必须是唯一的，不能与其他表或视图同名。

<SELECT语句>：指定创建视图的 SELECT 语句，可用于查询多个基础表或源视图。

对于创建视图中的 SELECT 语句的指定存在以下限制：

用户除了拥有 CREATE VIEW 权限外，还具有操作中涉及的基础表和其他视图的相关权限。

SELECT 语句不能引用系统或用户变量。

SELECT 语句不能包含 FROM 子句中的子查询。

SELECT 语句不能引用预处理语句参数。

视图定义中引用的表或视图必须存在。但是，创建完视图后，可以删除定义引用的表或视图。可使用 CHECK TABLE 语句检查视图定义是否存在这类问题。  
 视图定义中允许使用 ORDER BY 语句，但是若从特定视图进行选择，而该视图使用了自己的 ORDER BY 语句，则视图定义中的 ORDER BY 将被忽略。  
 视图定义中不能引用 TEMPORARY 表（临时表），不能创建 TEMPORARY 视图。  
 WITH CHECK OPTION 的意思是，修改视图时，检查插入的数据是否符合 WHERE 设置的条件。

#### 修改视图

可以使用 ALTER VIEW 语句来对已有的视图进行修改。  
  
 语法格式如下：

ALTER VIEW <视图名> AS <SELECT语句>

语法说明如下：

<视图名>：指定视图的名称。该名称在数据库中必须是唯一的，不能与其他表或视图同名。

<SELECT 语句>：指定创建视图的 SELECT 语句，可用于查询多个基础表或源视图。

需要注意的是，对于 ALTER VIEW 语句的使用，需要用户具有针对视图的 CREATE VIEW 和 DROP 权限，以及由 SELECT 语句选择的每一列上的某些权限。  
  
修改视图的定义，除了可以通过 ALTER VIEW 外，也可以使用 DROP VIEW 语句先删除视图，再使用 CREATE VIEW 语句来实现。

#### 删除视图

可以使用 DROP VIEW 语句来删除视图。  
 语法格式如下：

DROP VIEW <视图名1> [ , <视图名2> …]

其中：<视图名>指定要删除的视图名。DROP VIEW 语句可以一次删除多个视图，但是必须在每个视图上拥有 DROP 权限。

#### 查看视图

查看视图的字段信息与查看数据表的字段信息一样，都是使用 DESCRIBE 关键字来查看的。具体语法如下：

DESCRIBE 视图名;

或简写成：

DESC 视图名;

### 存储过程

1. 什么是存储过程
2. 存储过程基本操作

#### 创建存储过程

可以使用 CREATE PROCEDURE 语句创建存储过程，语法格式如下：

CREATE PROCEDURE <过程名> ( [过程参数[,…] ] ) <过程体>  
 [过程参数[,…] ] 格式  
 [ IN | OUT | INOUT ] <参数名> <类型>

语法说明如下：

1) 过程名

存储过程的名称，默认在当前数据库中创建。若需要在特定数据库中创建存储过程，则要在名称前面加上数据库的名称，即 db\_name.sp\_name。  
 需要注意的是，名称应当尽量避免选取与 MySQL 内置函数相同的名称，否则会发生错误。

2) 过程参数

存储过程的参数列表。其中，<参数名>为参数名，<类型>为参数的类型（可以是任何有效的 MySQL 数据类型）。当有多个参数时，参数列表中彼此间用逗号分隔。存储过程可以没有参数（此时存储过程的名称后仍需加上一对括号），也可以有 1 个或多个参数。  
  
 MySQL 存储过程支持三种类型的参数，即输入参数、输出参数和输入/输出参数，分别用 IN、OUT 和 INOUT 三个关键字标识。其中，输入参数可以传递给一个存储过程，输出参数用于存储过程需要返回一个操作结果的情形，而输入/输出参数既可以充当输入参数也可以充当输出参数。  
 需要注意的是，参数的取名不要与数据表的列名相同，否则尽管不会返回出错信息，但是存储过程的 SQL 语句会将参数名看作列名，从而引发不可预知的结果。

3) 过程体

存储过程的主体部分，也称为存储过程体，包含在过程调用的时候必须执行的 SQL 语句。这个部分以关键字 BEGIN 开始，以关键字 END 结束。若存储过程体中只有一条 SQL 语句，则可以省略 BEGIN-END 标志。  
 在存储过程的创建中，经常会用到一个十分重要的 MySQL 命令，即 DELIMITER 命令，特别是对于通过命令行的方式来操作 MySQL 数据库的使用者，更是要学会使用该命令。  
 在 MySQL 中，服务器处理 SQL 语句默认是以分号作为语句结束标志的。然而，在创建存储过程时，存储过程体可能包含有多条 SQL 语句，这些 SQL 语句如果仍以分号作为语句结束符，那么 MySQL 服务器在处理时会以遇到的第一条 SQL 语句结尾处的分号作为整个程序的结束符，而不再去处理存储过程体中后面的 SQL 语句，这样显然不行。  
  
 为解决以上问题，通常使用 DELIMITER 命令将结束命令修改为其他字符。语法格式如下：

DELIMITER $$

语法说明如下：

$$ 是用户定义的结束符，通常这个符号可以是一些特殊的符号，如两个“?”或两个“￥”等。

当使用 DELIMITER 命令时，应该避免使用反斜杠“\”字符，因为它是 MySQL 的转义字符。

在 MySQL 命令行客户端输入如下 SQL 语句。

mysql > DELIMITER ??

成功执行这条 SQL 语句后，任何命令、语句或程序的结束标志就换为两个问号“??”了。  
 若希望换回默认的分号“;”作为结束标志，则在 MySQL 命令行客户端输入下列语句即可：

mysql > DELIMITER ;

注意：DELIMITER 和分号“;”之间一定要有一个空格。在创建存储过程时，必须具有 CREATE ROUTINE 权限。

#### 修改存储过程

MySQL 中通过 ALTER PROCEDURE 语句来修改存储过程。本节将详细讲解修改存储过程的方法。  
  
 MySQL 中修改存储过程的语法格式如下：

ALTER PROCEDURE 存储过程名 [ 特征 ... ]

特征指定了存储过程的特性，可能的取值有：

CONTAINS SQL 表示子程序包含 SQL 语句，但不包含读或写数据的语句。

NO SQL 表示子程序中不包含 SQL 语句。

READS SQL DATA 表示子程序中包含读数据的语句。

MODIFIES SQL DATA 表示子程序中包含写数据的语句。

SQL SECURITY { DEFINER |INVOKER } 指明谁有权限来执行。

DEFINER 表示只有定义者自己才能够执行。

INVOKER 表示调用者可以执行。

COMMENT 'string' 表示注释信息。

#### 删除存储过程

存储过程被创建后，就会一直保存在数据库服务器上，直至被删除。当 [MySQL](http://c.biancheng.net/mysql/" \t "http://c.biancheng.net/view/_blank) 数据库中存在废弃的存储过程时，我们需要将它从数据库中删除。  
  
 MySQL 中使用 DROP PROCEDURE 语句来删除数据库中已经存在的存储过程。语法格式如下：

DROP PROCEDURE [ IF EXISTS ] <过程名>

语法说明如下：

过程名：指定要删除的存储过程的名称。

IF EXISTS：指定这个关键字，用于防止因删除不存在的存储过程而引发的错误。

注意：存储过程名称后面没有参数列表，也没有括号，在删除之前，必须确认该存储过程没有任何依赖关系，否则会导致其他与之关联的存储过程无法运行。

#### 查看存储过程

MySQL 中可以通过 SHOW STATUS 语句查看存储过程的状态，其基本语法形式如下：

SHOW PROCEDURE STATUS LIKE 存储过程名;

LIKE 存储过程名用来匹配存储过程的名称，LIKE 不能省略。

### 游标

#### 声明游标

MySQL 中使用 DECLARE 关键字来声明游标，并定义相应的 SELECT 语句，根据需要添加 WHERE 和其它子句。其语法的基本形式如下：

DECLARE cursor\_name CURSOR FOR select\_statement;

其中，cursor\_name 表示游标的名称；select\_statement 表示 SELECT 语句，可以返回一行或多行数据。

#### 打开游标

声明游标之后，要想从游标中提取数据，必须首先打开游标。在 MySQL 中，打开游标通过 OPEN 关键字来实现，其语法格式如下：

OPEN cursor\_name;

其中，cursor\_name 表示所要打开游标的名称。需要注意的是，打开一个游标时，游标并不指向第一条记录，而是指向第一条记录的前边。  
 在程序中，一个游标可以打开多次。用户打开游标后，其他用户或程序可能正在更新数据表，所以有时会导致用户每次打开游标后，显示的结果都不同。

#### 使用游标

游标顺利打开后，可以使用 FETCH...INTO 语句来读取数据，其语法形式如下：

FETCH cursor\_name INTO var\_name [,var\_name]...

上述语句中，将游标 cursor\_name 中 SELECT 语句的执行结果保存到变量参数 var\_name 中。变量参数 var\_name 必须在游标使用之前定义。使用游标类似高级语言中的数组遍历，当第一次使用游标时，此时游标指向结果集的第一条记录。  
 MySQL 的游标是只读的，也就是说，你只能顺序地从开始往后读取结果集，不能从后往前，也不能直接跳到中间的记录。

4. 关闭游标

游标使用完毕后，要及时关闭，在 MySQL 中，使用 CLOSE 关键字关闭游标，其语法格式如下：

CLOSE cursor\_name;

CLOSE 释放游标使用的所有内部内存和资源，因此每个游标不再需要时都应该关闭。  
 在一个游标关闭后，如果没有重新打开，则不能使用它。但是，使用声明过的游标不需要再次声明，用 OPEN 语句打开它就可以了。  
  
如果你不明确关闭游标，MySQL 将会在到达 END 语句时自动关闭它。游标关闭之后，不能使用 FETCH 来使用该游标。

### 绑定变量

#### 定义变量

MySQL 中可以使用 DECLARE 关键字来定义变量，其基本语法如下：

DECLARE var\_name[,...] type [DEFAULT value]

其中：

DECLARE 关键字是用来声明变量的；

var\_name 参数是变量的名称，这里可以同时定义多个变量；

type 参数用来指定变量的类型；

DEFAULT value 子句将变量默认值设置为 value，没有使用 DEFAULT 子句时，默认值为 NULL。

例 1

下面定义变量 my\_sql，数据类型为 INT 类型，默认值为 10。SQL 语句如下：

DECLARE my\_sql INT DEFAULT 10;

#### 为变量赋值

MySQL 中可以使用 SET 关键字来为变量赋值，SET 语句的基本语法如下：

SET var\_name = expr[,var\_name = expr]...

其中：

SET 关键字用来为变量赋值；

var\_name 参数是变量的名称；

expr 参数是赋值表达式。

### 自定义函数

在 MySQL 中，使用 CREATE FUNCTION 语句来创建存储函数，其语法形式如下：

CREATE FUNCTION sp\_name ([func\_parameter[...]])  
 RETURNS type  
 [characteristic ...] routine\_body

其中：

sp\_name 参数：表示存储函数的名称；

func\_parameter：表示存储函数的参数列表；

RETURNS type：指定返回值的类型；

characteristic 参数：指定存储函数的特性，该参数的取值与存储过程是一样的；

routine\_body 参数：表示 SQL 代码的内容，可以用 BEGIN...END 来标示 SQL 代码的开始和结束。  
 注意：在具体创建函数时，函数名不能与已经存在的函数名重名。除了上述要求外，推荐函数名命名（标识符）为 function\_xxx 或者 func\_xxx。  
 func\_parameter 可以由多个参数组成，其中每个参数由参数名称和参数类型组成，其形式如下：

[IN | OUT | INOUT] param\_name type;

其中：

IN 表示输入参数，OUT 表示输出参数，INOUT 表示既可以输入也可以输出；

param\_name 参数是存储函数的参数名称；

type 参数指定存储函数的参数类型，该类型可以是 MySQL 数据库的任意数据类型。

### 触发器

1. 什么是触发器
2. 触发器基本操作：

#### 创建触发器

在 MySQL 5.7 中，可以使用 CREATE TRIGGER 语句创建触发器。  
  
 语法格式如下：

CREATE <触发器名> < BEFORE | AFTER >  
<INSERT | UPDATE | DELETE >  
ON <表名> FOR EACH Row<触发器主体>

语法说明如下。

1) 触发器名

触发器的名称，触发器在当前数据库中必须具有唯一的名称。如果要在某个特定数据库中创建，名称前面应该加上数据库的名称。

2) INSERT | UPDATE | DELETE

触发事件，用于指定激活触发器的语句的种类。  
  
注意：三种触发器的执行时间如下。

INSERT：将新行插入表时激活触发器。例如，INSERT 的 BEFORE 触发器不仅能被 MySQL 的 INSERT 语句激活，也能被 LOAD DATA 语句激活。

DELETE： 从表中删除某一行数据时激活触发器，例如 DELETE 和 REPLACE 语句。

UPDATE：更改表中某一行数据时激活触发器，例如 UPDATE 语句。

3) BEFORE | AFTER

BEFORE 和 AFTER，触发器被触发的时刻，表示触发器是在激活它的语句之前或之后触发。若希望验证新数据是否满足条件，则使用 BEFORE 选项；若希望在激活触发器的语句执行之后完成几个或更多的改变，则通常使用 AFTER 选项。

4) 表名

与触发器相关联的表名，此表必须是永久性表，不能将触发器与临时表或视图关联起来。在该表上触发事件发生时才会激活触发器。同一个表不能拥有两个具有相同触发时刻和事件的触发器。例如，对于一张数据表，不能同时有两个 BEFORE UPDATE 触发器，但可以有一个 BEFORE UPDATE 触发器和一个 BEFORE INSERT 触发器，或一个 BEFORE UPDATE 触发器和一个 AFTER UPDATE 触发器。

5) 触发器主体

触发器动作主体，包含触发器激活时将要执行的 MySQL 语句。如果要执行多个语句，可使用 BEGIN…END 复合语句结构。

6) FOR EACH ROW

一般是指行级触发，对于受触发事件影响的每一行都要激活触发器的动作。例如，使用 INSERT 语句向某个表中插入多行数据时，触发器会对每一行数据的插入都执行相应的触发器动作。

注意：每个表都支持 INSERT、UPDATE 和 DELETE 的 BEFORE 与 AFTER，因此每个表最多支持 6 个触发器。每个表的每个事件每次只允许有一个触发器。单一触发器不能与多个事件或多个表关联。

另外，在 MySQL 中，若需要查看数据库中已有的触发器，则可以使用 SHOW TRIGGERS 语句。

#### 删除触发器

与其他 [MySQL](http://c.biancheng.net/mysql/" \t "http://c.biancheng.net/view/_blank) 数据库对象一样，可以使用 DROP 语句将触发器从数据库中删除。  
  
 语法格式如下：

DROP TRIGGER [ IF EXISTS ] [数据库名] <触发器名>

语法说明如下：

1) 触发器名

要删除的触发器名称。

2) 数据库名

可选项。指定触发器所在的数据库的名称。若没有指定，则为当前默认的数据库。

3) 权限

执行 DROP TRIGGER 语句需要 SUPER 权限。

4) IF EXISTS

可选项。避免在没有触发器的情况下删除触发器。

注意：删除一个表的同时，也会自动删除该表上的触发器。另外，触发器不能更新或覆盖，为了修改一个触发器，必须先删除它，再重新创建。

#### 查看触发器

在 MySQL 中，可以通过 SHOW TRIGGERS 语句来查看触发器的基本信息，语法格式如下：

SHOW TRIGGERS;

在 MySQL 中，所有触发器的信息都存在 information\_schema 数据库的 triggers 表中，可以通过查询命令 SELECT 来查看，具体的语法如下：

SELECT \* FROM information\_schema.triggers WHERE trigger\_name= '触发器名';

其中，'触发器名'用来指定要查看的触发器的名称，需要用单引号引起来。这种方式可以查询指定的触发器，使用起来更加方便、灵活。

### 全文索引

### SQLMode

1. 什么是SQL Mode？
2. 常见SQL Mode
3. 数据迁移中使用SQL Mode

# 进阶

了解基本SQL操作和高级特性可以基本满足平时开发的需要，但是如果真正从事数据库底层开发，还需要了解这些SQL语句背后的算法，这就涉及索引实现及优化，锁的使用，事务控制，SQL优化。

如果需要掌握一个庞大的数据库系统的操作和运维，还需要掌握主从复制、备份恢复、高可用、基准测试。

## 索引

1. 索引基本操作
2. 索引底层数据结构
3. 索引失效
4. 索引设计原则及优化

## 锁

1. MySQL锁的分类
2. 行锁的实现算法
3. 锁与并发控制
4. InnoDB死锁

## 日志

1. 错误日志
2. 二进制日志
3. 查询日志
4. 慢查询日志
5. Redo log、undo log

## 备份恢复

1. 备份恢复策略
2. 逻辑备份与恢复
3. 物理备份与恢复
4. 表的导入导出

## 事务

1. 什么是事务？
2. ACID实现原理
3. 事务控制
4. 分布式事务

## SQL优化

1. SQL优化的基本步骤
2. 常用优化策略
3. 服务器优化
4. 磁盘存储优化
5. 应用优化

## 架构

### 主从复制

1. 主从复制原理
2. 启动参数
3. 主从切换

### MySQL Cluster

### 高可用

1、高可用架构

## 词法解析/语法解析

## InnoDB存储引擎

1. 存储引擎概述
2. 体系架构
3. Checkpoint
4. Double Write
5. 自适应哈希
6. 异步IO
7. 表结构与行记录
8. SQL语句算法

## 测试

### 基准测试

### 测试工具

# 拓展

## Redis

1. 什么是NoSQL？
2. 数据类型
3. 基本操作
4. 持久化
5. 事务
6. 主从复制
7. 集群

## MongoDB

1. 索引
2. 事务

## Hive SQL

# 学习资料

## 推荐书籍

《SQL必知必会》

## 技术博客

## 教学视频