# 操作

## 1、数据库操作

（1）创建库：CREATE DATABASE[ IF NOT EXISTS] 数据库名 数据库选项

数据库选项： CHARACTER SET charset\_name

COLLATE collation\_name

（2）删除库：DROP DATABASE[ IF EXISTS] 数据库名

同时删除该数据库相关的目录及其目录内容

（3）修改库的选项信息：ALTER DATABASE 库名 选项信息

（4）查看已有库： SHOW DATABASES[LIKE 'PATTERN']

（5）查看当前数据库：SELECT DATABASE();

（6）查看当前库信息：SHOW CREATE DATABASE 数据库名

（7）显示当前时间、用户名、数据库版本：SELECT now(), user(), version();

## 2、表操作

（1）创建表：CREATE [TEMPORARY] TABLE[ IF NOT EXISTS] [库名.]表名 ( 表的结构定义 )[ 表选项]

字段定义(每个字段必须有数据类型, 最后一个字段后不能有逗号)：

字段名 数据类型 [NOT NULL | NULL] [DEFAULT default\_value] [AUTO\_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY] [COMMENT 'string']

字符集：CHARSET = charset\_name (如果表没有设定，则使用数据库字符集)

存储引擎：ENGINE = engine\_name

自增起始数：AUTO\_INCREMENT = 行数

表选项： 数据文件目录：DATA DIRECTORY = '目录'

索引文件目录：INDEX DIRECTORY = '目录'

表注释：COMMENT = 'string'

分区选项：PARTITION BY ... (详细见手册)

（2）删除表：DROP TABLE[ IF EXISTS] 表名 ...

（3）清空表数据：TRUNCATE [TABLE] 表名 (删除表再创建，重置auto\_increment的值，分区表会保留分区)

（4）修改表本身的选项：ALTER TABLE 表名 表的选项 (eg: ALTER TABLE 表名 ENGINE=MYISAM;)

（5）对表进行重命名： RENAME TABLE 原表名 TO 新表名

RENAME TABLE 原表名 TO 库名.表名 （可将表移动到另一个数据库）

（6）修改表的字段结构：ALTER TABLE 表名 操作名

增加字段：ADD[ COLUMN] 字段定义

删除字段：DROP[ COLUMN] 字段名

修改字段属性：MODIFY[ COLUMN] 字段名 字段属性 (不能修改字段名,所有原有属性也需写上)

修改字段名：CHANGE[ COLUMN] 原字段名 新字段名 字段属性

操作名 创建主键：ADD PRIMARY KEY(字段名)

创建唯一索引：ADD UNIQUE [索引名] (字段名)

创建普通索引：ADD INDEX [索引名] (字段名)

删除主键：DROP PRIMARY KEY -- (删除主键前需删除其AUTO\_INCREMENT属性)

删除索引：DROP INDEX 索引名

删除外键DROP FOREIGN KEY 外键

（7）复制表结构：CREATE TABLE 表名 LIKE 要复制的表名

（8）复制表结构和数据：CREATE TABLE 表名 [AS] SELECT \* FROM 要复制的表名

（9）优化表：OPTIMIZE [LOCAL | NO\_WRITE\_TO\_BINLOG] TABLE [tbl\_name] ...

（10）修复表: REPAIR [LOCAL|NO\_WRITE\_TO\_BINLOG] TABLE [tbl\_name] ... [QUICK] [EXTENDED] [USE\_FRM]

（11）查看所有表： SHOW TABLES[ LIKE 'pattern']

SHOW TABLES FROM 库名

（12）查看表结构： SHOW CREATE TABLE 表名 (信息更详细)

DESC 表名 / DESCRIBE 表名 / EXPLAIN 表名 / SHOW COLUMNS FROM 表名 [LIKE 'PATTERN']

SHOW TABLE STATUS [FROM db\_name] [LIKE 'pattern']

（13）检查表是否有错误：CHECK TABLE [ tbl\_name] ... [option] ...

（14）分析表：ANALYZE [LOCAL | NO\_WRITE\_TO\_BINLOG] TABLE [tbl\_name] ...

## 3、数据操作

（1）增： INSERT [INTO] 表名 [(字段列表)] VALUES [(值列表), (值列表), ...]

INSERT [INTO] 表名 SET 字段名=值[, 字段名=值, ...]

REPLACE：存在则update，不存在则insert。

（2）删：DELETE FROM 表名[ 删除条件子句] (没有条件子句，则会删除全部)

（3）改：UPDATE 表名 SET 字段名=新值[, 字段名=新值] [更新条件]

（4）查： SELECT [ALL|DISTINCT] \* FROM 表名 WHERE

-> GROUP BY [分组字段 升序ASC/降序DESC]

-> HAVING

-> ORDER BY [排序字段升序ASC/降序DESC]...

-> LIMIT [起始位置, 获取条数]

count()、sum()、max()、min()、avg()、group\_concat()等合计函数需配合GROUP BY使用

HAVING：与 where 功能、用法相同，执行时机不同。

where 在开始时执行检测数据，对原数据进行过滤。having 对筛选出的结果再次进行过滤。

where 字段必须是数据表存在的，having 字段必须是查询出来的。

where 不可以使用字段的别名，having 可以。因为执行WHERE代码时，可能尚未确定列值。

where 不可以使用合计函数。一般需用合计函数才会用 having

SQL标准要求HAVING必须引用GROUP BY子句中的列或用于合计函数中的列。

（5）合并多个查询：Union /Union All

（6）连接查询：inner join/ left join / right join

## 4、SQL编程

（1）变量申明： declare var\_name[,...] type [default value]

（2）变量赋值： set @var = value

select value into @var

注：为了避免=被当作关系运算符看待，可使用:=代替

（3）if语句：if condition then result

elseif condition then result

else result

end if

（4）case语句： CASE value WHEN [condition ] THEN result

[WHEN [condition] THEN result ...]

[ELSE result]

END

（5）while循环： [begin\_label:] while search\_condition do

statement\_list

end while [end\_label];

注：标签需要成对出现

退出循环 退出整个循环 leave

退出当前循环 iterate

通过退出的标签决定退出哪个循环

（6）内置函数：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数值函数 | abs(x) | 绝对值 abs(-10.9) = 10 | mod(m, n) | m%n m mod n 求余 |
| ceil(x) | 向上取整 ceil(10.1) = 11 | pi() | 获得圆周率 |
| floor(x) | 向下取整 floor (10.1) = 10 | pow(m, n) | m^n |
| round(x) | 四舍五入去整 | sqrt(x) | 算术平方根 |
| truncate(x, d) | 截取d位小数 | rand() | 随机数 |
| format(x, d) | | 格式化千分位数值 format(34567.456, 2) = 34,567.46 | |
| 时间日期函数 | now(), current\_timestamp() | 当前日期时间 | current\_date() | 当前日期 |
| date('yyyy-mm-dd hh:ii:ss') | 获取日期部分 | current\_time() | 当前时间 |
| time('yyyy-mm-dd hh:ii:ss') | 获取时间部分 | unix\_timestamp() | 获得unix时间戳 |
| date\_format('yyyy-mm-dd hh:ii:ss', '%d %y %a %d %m %b %j') | 格式化时间 | from\_unixtime() | 从时间戳获得时间 |
| 字符串函数 | length(string) | string长度，字节 | concat(string [,...]) | 连接字串 |
| char\_length(string) | string的字符个数 | repeat(string, count) | 重复count次 |
| load\_file(file\_name) | 从文件读取内容 | charset(str) | 返回字串字符集 |
| ltrim(string) | 去除前端空格 | lcase(string) | 转换成小写 |
| rtrim(string) | 去除后端空格 | strcmp(string1 ,string2) | 逐字符比较两字串大小 |
| left(string, length) | | 从string2中的左边起取length个字符 | |
| substring(str, index[,length]) | | 从str的index开始,取length个字符 | |
| replace(str ,search\_str ,replace\_str) | | 在str中用replace\_str替换search\_str | |
| instr(string ,substring) | | 返回substring首次在string中出现的位置 | |
| locate(substring, string [,start\_position]) | | 同instr,但可指定开始位置 | |
| lpad(string, length, pad) | | 重复用pad加在string开头,直到字串长度为length | |
| rpad(string, length, pad) | | 在str后用pad补充,直到长度为length | |
| 聚合函数 | sum() | 求和 | min() | 求最小值 |
| max() | 求最大值 | avg() | 求平均值 |
| count() | 返回不同的非NULL值数目 |  |  |
| group\_concat() | 组内字符串连接(返回带有来自一个组的连接的非NULL值的字符串结果) | | |
| 其他常用函数 | md5() |  | default() |  |
| if(expr1,expr2,expr3) | | expr1为true，则返回expr2，否则返回expr3 | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 5、自定义函数

（1）新建：CREATE FUNCTION function\_name (参数列表) RETURNS 返回值类型

Begin

函数体(含有return 返回值语句)

End

（2）删除：DROP FUNCTION [IF EXISTS] function\_name;

（3）查看：SHOW FUNCTION STATUS LIKE 'partten'

SHOW CREATE FUNCTION function\_name;

（4）修改：ALTER FUNCTION function\_name 函数选项

## 6、存储过程

语法：CREATE PROCEDURE 过程名 (参数列表)

BEGIN

过程体

END

参数列表：IN|OUT|INOUT 参数名 数据类型

IN 输入：在调用过程中，将数据输入到过程体内部的参数

OUT 输出：在调用过程中，将过程体处理完的结果返回到客户端

INOUT 输入输出：既可输入，也可输出

调用：CALL 过程名

/\* 触发器 \*/ ------------------

触发程序是与表有关的命名数据库对象，当该表出现特定事件时，将激活该对象

监听：记录的增加、修改、删除。

-- 创建触发器

CREATE TRIGGER trigger\_name trigger\_time trigger\_event ON tbl\_name FOR EACH ROW trigger\_stmt

参数：

trigger\_time是触发程序的动作时间。它可以是 before 或 after，以指明触发程序是在激活它的语句之前或之后触发。

trigger\_event指明了激活触发程序的语句的类型

INSERT：将新行插入表时激活触发程序

UPDATE：更改某一行时激活触发程序

DELETE：从表中删除某一行时激活触发程序

tbl\_name：监听的表，必须是永久性的表，不能将触发程序与TEMPORARY表或视图关联起来。

trigger\_stmt：当触发程序激活时执行的语句。执行多个语句，可使用BEGIN...END复合语句结构

-- 删除

DROP TRIGGER [schema\_name.]trigger\_name

可以使用old和new代替旧的和新的数据

更新操作，更新前是old，更新后是new.

删除操作，只有old.

增加操作，只有new.

-- 注意

1. 对于具有相同触发程序动作时间和事件的给定表，不能有两个触发程序。

-- 字符连接函数

concat(str1,str2,...])

concat\_ws(separator,str1,str2,...)

-- 分支语句

if 条件 then

执行语句

elseif 条件 then

执行语句

else

执行语句

end if;

-- 修改最外层语句结束符

delimiter 自定义结束符号

SQL语句

自定义结束符号

delimiter ; -- 修改回原来的分号

-- 语句块包裹

begin

语句块

end

-- 特殊的执行

1. 只要添加记录，就会触发程序。

2. Insert into on duplicate key update 语法会触发：

如果没有重复记录，会触发 before insert, after insert;

如果有重复记录并更新，会触发 before insert, before update, after update;

如果有重复记录但是没有发生更新，则触发 before insert, before update

3. Replace 语法 如果有记录，则执行 before insert, before delete, after delete, after insert

索引

事务

特性ACID

隔离级别

并发事务导致哪些问题

优化

大表优化：限定数据范围、读写分离、垂直分区、水平分区

锁