## 歪麦博客摘抄

摘抄地址：<https://www.awaimai.com/534.html>

### 1 常用命令

第一步，连接数据库。

$ mysql -u root -p          # 进入MySQL bin目录后执行，回车后输入密码连接。

# 常用参数：-h 服务器地址，-u 用户名，-p 密码，-P 端口

然后是一些常用的命令。  
数据库操作：

> create database dbname;          # 创建数据库，数据库名为dbname

> CREATE DATABASE `todo` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci; # 创建todo数据库，并指定字符集

> show databases; # 显示所有数据库

> alter database character set utf8; # 修改数据库字符集

> use dbname;          # 选择数据库

> status;           # 查看当前选择（use）的数据库

> drop database dbname;          # 删除数据库

数据表操作：

> show tables;           # 显示所有表

> describe tablename;          # 表结构详细描述

> desc tablename; # 同 describe 命令一样

> create table newtable like oldtable; # 复制表结构

> insert into newtable select \* from oldtable; #复制表数据

> rename table tablelname to new\_tablelname # 重命名表，同时命名多个表用逗号“,”分割

> drop table tablename;           # 删除表

界面操作：

> select version(),current\_date;          # 显示当前mysql版本和当前日期

> exit;           # 返回Shell命令行

### 2 创建数据表

create table命令用来创建数据表，格式：

create table <表名> (<字段名1> <类型1> [,..<字段名n> <类型n>]);

数据的类型之后还可以包含：数据宽度 + 是否为空 + 是否主键 + 自动增加 + 默认值，它们书写时不限顺序。但是字段名和字段类型必须是第一和第二位。例如，建立一个名为user的表：

mysql> create table user(

    -> id int(4) not null primary key auto\_increment,

    -> name char(20) not null,

    -> sex int(4) not null default '0',

    -> degree double(16,2)

-> );

### 3 修改root的密码

$ mysqladmin -u root password # 原始密码为空的情况New password: <输入新的密码>

Confirm new password: <再次输入新密码>

$ mysqladmin -u root -p password # 原始密码不为空的情况

Enter password: <输入旧的密码>New password: <输入新的密码>

Confirm new password: <再次输入新密码>

$ mysqladmin -uroot -p123456 password # 原始密码不为空的情况，效果和第二种方法一样，只是显式的输入了原始密码New password: <输入新的密码>

Confirm new password: <再次输入新密码>

### 4 备份和恢复

$ mysqldump -u root -p dbname > dbname.sql # 备份整个数据库（包含表结构和数据）

$ mysqldump -u root -p -d dbname > dbname.sql # 备份数据库表结构，不包含数据，

$ mysqldump -u root -p dbname tablename > tablename.sql # 备份数据库中的某张数据表（包含表结构和数据）

$ mysqldump -u root -p dbname tablename1 tablename2 > tables.sql # 备份数据库中2张数据表

$ mysqldump -u root -p -d dbname tablename > tablename.sql # 备份数据库中的某张数据表的表结构（不含数据）

$ mysqladmin -u root -p create dbname # 恢复数据库步骤1：创建数据库

$ mysql -u root -p dbname < dbname.sql # 恢复数据库步骤2：恢复数据

如果是网络上的服务器，可以在mysqldump之后用-h指定服务器地址，例如：

$ mysqldump -h sql.domain.com -u root -p dbname > dbname.sql

导出查询结果到本地计算机：

mysql -h127.0.0.1 -P3306 -uroot -proot -Ae "use test;select \* from user where status=4 order by id desc;" > "C:\Users\Gary\user.txt"

### 5 用户和权限管理

MySQL 默认有个root用户，但是这个用户权限太大，一般只在管理数据库时候才用。如果在项目中要连接 MySQL 数据库，则建议新建一个权限较小的用户来连接。在 MySQL 命令行模式下输入如下命令可以为 MySQL 创建一个新用户：

> CREATE USER username IDENTIFIED BY '123456'; # 其中，username是用户名，123456是用户密码

新用户创建完成，但是此刻如果以此用户登陆的话，会报错，因为我们还没有为这个用户分配相应权限，分配权限的命令是grant，格式为：

GRANT <权限> ON <数据库>.<表名> TO <用户名>@<登录主机> IDENTIFIED BY "<密码>"

其中，权限可以是：all、select、insert、delete、update、drop等值。

于是，用下面的命令就可以执行特定的功能：

> GRANT ALL ON \*.\* TO 'username'@'localhost' IDENTIFIED BY '123456'; # 给用户 username 分配所有数据库的所有权限

> REVOKE ALL ON \*.\* FROM 'username'@'localhost'; # 如果觉得上面的权限太大，用 REVOKE 删除原来权限

> GRANT ALL ON dbname.\* TO 'username'@'localhost' IDENTIFIED BY '123456'; # 重新授予仅在dbname数据库上的权限

> GRANT SELECT, UPDATE ON dbname.\* TO 'username'@'localhost' IDENTIFIED BY '123456'; # 仅授予select、update权限，无法执行insert、delete等命令

> FLUSH PRIVILEGES; # 每当调整权限后，通常需要用这个命令刷新权限

> DROP USER username@localhost; # 删除用户

仔细观察上面几个命令，可以发现不管是授权，还是删除授权，都要指定响应的host（即@符号后面的登录主机，主机也可以用%通配符表示所有主机，或者192.168.1.% 表示特定主机段）。因为以上及格命令实际上都是在操作 mysql 数据库中的user表，可以用如下命令查看相应用户及对应的host:

SELECT user, host FROM user;

当然，这个表中还包含很多其它例如用户密码、权限设置等很多内容，操作时候尤其需要小心。

SQL 语句主要可以划分为以下 3 个类别。

 DDL（Data Definition Languages）语句：数据定义语言，这些语句定义了不同的数据段、

数据库、表、列、索引等数据库对象的定义。常用的语句关键字主要包括 create、drop、alter

等。

 DML（Data Manipulation Language）语句：数据操纵语句，用于添加、删除、更新和查

询数据库记录，并检查数据完整性，常用的语句关键字主要包括 insert、delete、udpate 和

select 等。

 DCL（Data Control Language）语句：数据控制语句，用于控制不同数据段直接的许可和

访问级别的语句。这些语句定义了数据库、表、字段、用户的访问权限和安全级别。主要的

语句关键字包括 grant、revoke 等。

# MySQL

# 基础篇

## 常用命令

**Mysql -uroot -proot #链接数据库**

**杂项**

**Show warnings; #当SQL语句插入成功，却有warning提醒，可用此查看详情**

**show create table emp \G; #查看详细的数据表结构（即创建数据表的sql语句）**

**show table status like 'city' \G #查看innodb表中的外键信息**

“\G”选项的含义是使得记录能够按照字段竖着排列，对于内容比较长的记录更易于显示。

## 数据库操作命令（DDL）

information\_schema：主要存储了系统中的一些数据库对象信息。比如用户表信息、列信息、权限信息、字符集信息、分区信息等。

 cluster：存储了系统的集群信息。

 mysql：存储了系统的用户权限信息。

 test：系统自动创建的测试数据库，任何用户都可以使用。

-----------------------------------------------------------------数据库操作指令-------------------------------------------------------------------------

**create database name; #创建简单数据库**

**Show databases #查询所有的数据库**

**Use test; #打开数据库**

**Status; #查看当前选择（test）的数据库**

**Drop database name; #删除数据库**

-----------------------------------------------------------------数据表操作指令-------------------------------------------------------------------------

**Show tables; #查询当前数据库下所有的数据表**

**Desc name; #查看表结构**

**Describe name; #同desc 表名一样**

**show create table emp \G; #查看详细的数据表结构（即创建数据表的sql语句）**

**Drop table name # 删除数据表**

“\G”选项的含义是使得记录能够按照字段竖着排列，对于内容比较长的记录更易于显示。

-----------------------------------------------------------------其他操作指令----------------------------------------------------------------------------

**Exit; #退出数据库**

### （1）创建表

Create table命令用来创建数据表，格式：

数据的类型之后还可以包含：数据宽度 + 是否为空 + 是否主键 + 自动增加 + 默认值，它们的书写时不限顺序。但是字段名和字段类型必须是第一位和第二为。例如，建立一个名为user的表：

mysql> create table user(

    -> id int(4) not null primary key auto\_increment,

    -> name char(20) not null,

    -> sex int(4) not null default '0',

    -> degree double(16,2)

-> );

### 修改表

**修改表类型，语法：**

**ALTER TABLE tablename MODIFY [COLUMN] column\_definition [FIRST | AFTER col\_name]**

mysql> alter table user modify name varchar(20); #将user表中name的字段的类型修改；

**增加表字段，语法：**

**ALTER TABLE tablename ADD [COLUMN] column\_definition [FIRST | AFTER col\_name]**

mysql> alter table user add column age int(3); #在user表中新增一个age字段

**删除表字段，语法：**

**ALTER TABLE tablename DROP [COLUMN] col\_name**

mysql> alter table user drop column age; #将user表中的age字段删除

**字段改名，语法：**

**ALTER TABLE tablename CHANGE [COLUMN] old\_col\_name column\_definition [FIRST|AFTER col\_name]**

mysql> alter table user change age age2 int(4); #将user表中的age字段名修改，对应类型修改；

**修改字段排列顺序**

**前面介绍的的字段增加和修改语法（ADD/CNAHGE/MODIFY）中，都有一个可选项 first|after**

**column\_name，这个选项可以用来修改字段在表中的位置，默认 ADD 增加的新字段是加在**

**表的最后位置，而 CHANGE/MODIFY 默认都不会改变字段的位置。**

mysql> alter table user add birth date after name; #在user表中name字段后面新增一个birth字段

mysql> alter table user modify age int(3) first; #将user表中的age字段移动至最前

mysql> alter table user modify age int(3) after name; #将user表中的age字段移动至name字段后

**修改表名，语法：**

**ALTER TABLE tablename RENAME [TO] newtablena**

mysql> alter table user rename user1; #将user 数据表的表名修改为user1;

## DML语句（简单介绍）

### 新增操作

**插入记录语法：**

**INSERT INTO tablename (field1,field2,……fieldn) VALUES(value1,value2,……valuesn);**

mysql> insert into emp (ename,sal) values('dony',1000); #添加一条数据

mysql> insert into emp values('lisa','2003-02-01','3000',2);#同样添加一条数据

**插入多条数据，语法：**

**INSERT INTO tablename (field1, field2,……fieldn)**

**VALUES**

**(record1\_value1, record1\_value2,……record1\_valuesn),**

**(record2\_value1, record2\_value2,……record2\_valuesn),**

**……**

**(recordn\_value1, recordn\_value2,……recordn\_valuesn)**

**;**

mysql> insert into dept values(5,'dept5'),(6,'dept6'); #插入多条数据

### 更新操作

**更新语法：**

**UPDATE tablename SET field1=value1，field2.=value2，……fieldn=valuen [WHERE CONDITION]**

**更新多个表多条数据语法：**

**UPDATE t1,t2…tn set t1.field1=expr1,tn.fieldn=exprn [WHERE CONDITION]**

mysql> update emp a , dept b set a.sal=a.sal\*b.deptno , b.deptname=a.ename where a.deptno=b.deptno; #修改多条数据

### 删除操作

**删除语法：**

**DELETE FROM tablename [WHERE CONDITIO**

**删除多个表中的多条数据语法：**

**DELETE t1,t2…tn FROM t1,t2…tn [WHERE CONDITIO**

mysql> delete a,b from emp a, dept b where a.deptno=b.deptno and a.deptno=3;#删除多表中多条数据

### 查询操作

**查询的比较复杂，只用最基本的语法**：

**SELECT \* FROM tablename [WHERE CONDITIO**

**①查询不重复的记录 （使用关键字 DISTINCT）**

mysql> select distinct deptno from emp; #查询emp表中deptno字段值不重复的记录

**②条件查询 （使用关键字 WHERE）**

mysql> select \* from emp where deptno=1 #查询em表中deptno值为1的记录

**③排序和限制 语法： （使用关键字ORDER BY）**

**SELECT \* FROM tablename [WHERE CONDITION] [ORDER BY field1 [DESC|ASC] ， field2 [DESC|ASC]，……fieldn [DESC|ASC]] # DESC降序 ASC升序 默认升序**

mysql> select \* from emp order by deptno,sal desc; #倒序查询deptno,sal的记录

**④如果只想显示一部分，语法：（使用关键字 LIMIT）**

**SELECT …… [LIMIT offset\_start,row\_count] #offset\_start 表示记录的起始偏移量，row\_count 表示显示的行数。 #（其实就是开始的位置） （向后查询几条）**

mysql> select \* from emp order by sal limit 1,3; #升序查询sal字段从第一条开始向后三条

**⑤聚合查询 语法：**

**SELECT [field1,field2,……fieldn] fun\_name**

**FROM tablename**

**[WHERE where\_contition]**

**[GROUP BY field1,field2,……fieldn**

**[WITH ROLLUP]]**

**[HAVING where\_contition]**

**fun\_name 表示要做的聚合操作，也就是聚合函数，常用的有 sum（求和）、count(\*)（记**

**录数）、max（最大值）、min（最小值）。**

mysql> select count(1) from emp; #**COUNT**函数应用

mysql> select sum(sal),max(sal),min(sal) from emp; # **SUM、MAX、MIN**函数应用

**GROUP BY 关键字表示要进行分类聚合的字段，比如要按照部门分类统计员工数量，部门**

**就应该写在 group by 后面。**

mysql> select deptno,count(1) from emp group by deptno;

**WITH ROLLUP 是可选语法，表明是否对分类聚合后的结果进行再汇总。**

mysql> select deptno,count(1) from emp group by deptno with rollu;

**HAVING 关键字表示对分类后的结果再进行条件的过滤。**

mysql> select deptno,count(1) from emp group by deptno having count(1)>1;

### 表连接

**表连接分内连接和外连接**

**内连接：內连接仅选出两张表中互相匹配的记录（即内连接获取的是自己查询条件要求的汇集至一起）**

mysql> select ename,deptname from emp,dept where emp.deptno=dept.deptno;

**外连接：外连接会选出其他不匹配的记录**

**外连接分左连接与右连接 （左右连接相似，可互相转换）**

**（left join on）左连接：包含所有的左边表中的记录甚至是右边表中没有和它匹配**

mysql> select ename,deptname from emp left join dept on emp.deptno=dept.deptn

**（right join on）右连接：包含所有的右边表中的记录甚至是左边表中没有和它匹配的记录**

mysql> select ename,deptname from dept right join emp on dept.deptno=emp.deptn

### 子查询

**某些情况下，当我们查询的时候，需要的条件是另外一个 select 语句的结果**

**子查询的关键字主要包括 in、not in、=、!=、exists、not exists 等。**

**查询记录为多个时， 关键字（IN 、NOT IN）**

mysql> select \* from emp where deptno in(select deptno from dept);

**如果子查询记录唯一，可以用 “ =” 代替 IN**

mysql> select \* from emp where deptno = (select deptno from dept limit 1);

**当表连接代替子查询**

mysql> select emp.\* from emp ,dept where emp.deptno=dept.deptn

（注意：子查询和表连接之间的转换主要应用在两个方面：

 MySQL 4.1 以前的版本不支持子查询，需要用表连接来实现子查询的功能

 表连接在很多情况下用于优化子查询）

### 记录联合

**将两个表的数据按照一定的查询条件查询出来后，将结果合并到一起显示出来**

**关键字（UNION 和 UNION ALL ） ，语法：**

**SELECT \* FROM t1**

**UNION|UNION ALL**

**SELECT \* FROM t2**

**……**

**UNION|UNION ALL**

**SELECT \* FROM tn;**

**区别：**

**UNION ALL 是把结果集直接合并在一起，**

mysql> select deptno from emp

-> union all

> select deptno from dept;

**UNION 是将UNION ALL 后的结果进行一次 DISTINCT，去除重复记录后的结果。**

mysql> select deptno from emp

-> union

> select detno from dept;

## DCL语句

定义：DCL 语句主要是 DBA 用来管理系统中的对象权限时所使用，一般的开发人员很少

暂时简介（28章详讲）

## 数据类型

**每一个常量，变量和参数都有数据类型，它用来指定一定的存储格式、约束和有效范。**

**MySQL 提供了多种数据类型，主要包括数值型、字符串类型、日期和时间类型。**

### 数值类型



**①整数类型（TINYINT、SMALLINT、MEDIUMINT、INT、BIGINT）**

**所有的整数类型都有一个可选属性 UNSIGNED（无符号），**如果需要在字段里面保存非负数或者需要较大的上限值时可以用此选项，它的取值范围是正常值的下限取 0，上限取原值的 2 倍，

**整数类型还有一个属性AUTO\_INCREMENT。**在需要产生唯一标识符或顺序值时，可利用此属性，这个属性只用于整数类型。AUTO\_INCREMENT 值一般从 1 开始，每行增加 1。在插入 NULL 到一个 AUTO\_INCREMENT 列时，MySQL 插入一个比该列中当前最大值大 1 的值。一个表中最多只能有一个AUTO\_INCREMENT列。对于任何想要使用AUTO\_INCREMENT 的列，应该定义为 NOT NULL，并定义为 PRIMARY KEY 或定义为 UNIQUE 键。

**②浮点类型（FLOAT、DOUBLE）**

**浮点数如果不写精度和标度，则会按照实际精度值显示，如果有精度和标度，则会自动将四舍五入后的结果插入，系统不会报错；**

**FLOAT （单精度）**

**DOUBLE（双精度）**

**③订单类型 （DECIMAL）**

**定点数如果不写精度和标度，则按照默认值 decimal(10,0)来进行操作，并且如果数据超越了精度和标度值，系统则会报错**

**DECIMAL 更精确 适合货币等精度高数据**

**④位类型（BIT）（暂时用不到估计）**

**用于存放位字段值，BIT(M)可以用来存放多位二进制数，M 范围从 1～64，如果不写则默认为 1 位。对于位字段，直接使用 SELECT 命令将不会看到结果，可以用bin()（显示为二进制格式）或者 hex()（显示为十六进制格式）函数进行读取。**

### 日期时间类型



说明：

**如果要用来表示年月日，通常用 DATE 来表示。**

** 如果要用来表示年月日时分秒，通常用 DATETIME 表示。**

** 如果只用来表示时分秒，通常用 TIME 来表示。**

** 如果需要经常插入或者更新日期为当前系统时间，则通常使用 TIMESTAMP 来表示。TIMESTAMP 值返回后显示为“YYYY-MM-DD HH:MM:SS”格式的字符串，显示宽度固定为 19 个字符。如果想要获得数字值，应在 TIMESTAMP 列添加+0。**

** 如果只是表示年份，可以用 YEAR 来表示，它比 DATE 占用更少的空间。YEAR 有 2 位或4 位格式的年。默认是 4 位格式。在 4 位格式中，允许的值是 1901～2155 和 0000。在2 位格式中，允许的值是 70～69，表示从 1970～2069 年。MySQL 以 YYYY 格式显示 YEAR值**



**最常用的时间函数 （DATE、TIME、DATETIME）**

**DATETIME 是DATE与TIME的结合体**

**TIMESTAMP**

**TIMESTAMP支持的时间范围较小，其取值范围从19700101080001到2038年的某个时间，而DATETIME是从1000-01-01 00:00:00到9999-12-31 23:59:59，范围更大。**

** 表中的第一个TIMESTAMP列自动设置为系统时间。如果在一个TIMESTAMP列中插入NULL，则该列值将自动设置为当前的日期和时间。在插入或更新一行但不明确给TIMESTAMP列赋值时也会自动设置该列的值为当前的日期和时间，当插入的值超出取值范围时，MySQL认为该值溢出，使用“0000-00-00 00:00:00”进行填补。**

** TIMESTAMP的插入和查询都受当地时区的影响，更能反应出实际的日期。而DATETIME则只能反应出插入时当地的时区，其他时区的人查看数据必然会有误差的。**

** TIMESTAMP的属性受MySQL版本和服务器SQLMode的影响很大，本章都是以MySQL 5.0为例进行介绍，在不同的版本下可以参考相应的MySQL帮助文档**

**YEAR**

**YEAR 类型主要用来表示年份，当应用只需要记录年份时，用 YEAR 比 DATE 将更节省空间。**

**MySQL 以 YYYY 格式检索和显示 YEAR 值，范围是 1901～2155。当使用两位字符串表示**

**年份时，其范围为“00”到“99”。**

**“00”到“69”范围的值被转换为 2000～2069 范围的 YEAR 值**

**“70”到“99”范围的值被转换为 1970～1999 范围的 YEAR 值**

**总结**

**什么样的格式才能够正确地插入到对应的日期字段中呢？（以DATETIME为例）**

**YYYY-MM-DD HH:MM:SS 或 YY-MM-DD HH:MM:SS 格式的字符串。允许“不严格”**

**语法：任何标点符都可以用做日期部分或时间部分之间的间割符。例如，“98-12-31**

**11:30:45”、“98.12.31 11+30+45”、“98/12/31 11\*30\*45”和“98@12@31 11^30^45”**

**是等价的。对于包括日期部分间割符的字符串值，如果日和月的值小于 10，不需**

**要指定两位数。“1979-6-9”与“1979-06-09”是相同的。同样，对于包括时间部分**

**间割符的字符串值，如果时、分和秒的值小于 10，不需要指定两位数。“1979-10-30**

**1:2:3”与“1979-10-30 01:02:03”相同。**

** YYYYMMDDHHMMSS 或 YYMMDDHHMMSS 格式的没有间割符的字符串，假定字符**

**串对于日期类型是有意义的。例如，“19970523091528”和“970523091528”被解**

**释为“1997-05-23 09:15:28”，但“971122129015”是不合法的（它有一个没有意**

**义的分钟部分），将变为“0000-00-00 00:00:00”。**

** YYYYMMDDHHMMSS 或 YYMMDDHHMMSS 格式的数字，假定数字对于日期类型是**

**有意义的。例如，19830905132800和830905132800被解释为“1983-09-05 13:28:00”。**

**数字值应为 6、8、12 或者 14 位长。如果一个数值是 8 或 14 位长，则假定为**

**YYYYMMDD 或 YYYYMMDDHHMMSS 格式，前 4 位数表示年。如果数字 是 6 或 12**

**位长，则假定为 YYMMDD 或 YYMMDDHHMMSS 格式，前 2 位数表示年。其他数字**

**被解释为仿佛用零填充到了最近的长度。**

** 函数返回的结果，其值适合 DATETIME、DATE 或者 TIMESTAMP 上下文，例如 NOW()**

**或 CURRENT\_DATE。**

### 字符串类型



**CHAR 和VARCHAR**

**CHAR 和 VARCHAR 很类似，都用来保存 MySQL 中较短的字符串。主要区别在于存储方式的不同**

**CHAR**

**CHAR 列的长度固定为创建表时声明的长度，长度可以为从 0～255 的任何值；**

**在检索的时候，CHAR 列删除了尾部的空格**

**VARCHAR**

**VARCHAR 列中的值为可变长字符串，长度可以指定为 0～255（5.0.3 以前）或者 65535（5.0.3以后）之间的值。**

**在检索的时候， VARCHAR 则保留这些空格**

**BINARY 和 VARBINARY 类型（暂时了解）**

**BINARY 和 VARBINARY 类似于 CHAR 和 VARCHAR，不同的是它们包含二进制字符串而不包含非二进制字符串。**

**ENUM 类型**

**ENUM 中文名称叫枚举类型，它的值范围需要在创建表时通过枚举方式显式指定，对 1～255 个成员的枚举需要 1 个字节存储；对于 255～65535 个成员，需要 2 个字节存储。最多允许有 65535 个成员。**

**ENUM 类型只允许从值集合中选取单个值，而不能一次取多个值。**

**SET 类型**

**Set 和 ENUM 类型非常类似，也是一个字符串对象，里面可以包含 0～64 个成员。根据成员的不同，存储上也有所不同。**

**1～8 成员的集合，占 1 个字节。**

**9～16 成员的集合，占 2 个字节。**

**17～24 成员的集合，占 3 个字节。**

**25～32 成员的集合，占 4 个字节。**

**33～64 成员的集合，占 8 个字节。**

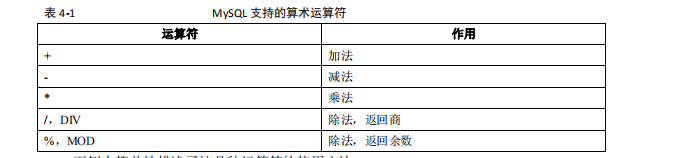
**Set 和 ENUM 除了存储之外，最主要的区别在于 Set 类型一次可以选取多个成员，而 ENUM则只能选一个。**

**（即 SET类型能一个字段中能够插入多个值，而ENUM中只能插入一个值）**

## 运算符

### （1）算术运算符

**MySQL 支持的算术运算符包括加、减、乘、除和模运算。它们是最常使用、最简单的一类运算符。**



 **+ 运算符用于获得一个或多个值的和。**

** - 运算符用于从一个值中减去另一个值。**

** \* 运算符使数字相乘,得到两个或多个值的乘积。**

** / 运算符用一个值除以另一个值得到商。**

**除法运算和模运算中，如果除数为 0，将是非法除数，返回结果为 NULL，**

**% 运算符用一个值除以另外一个值得到余数。**

**对于模运算，还有另外一种表达方式，使用 MOD(a,b)函数与 a%b 效果**

### （2）比较运算符

比较运算符可以用于比较数字、字符串和表达式。数字作为浮点数比较，而字符串以不区分大小写的方式进行比较。



“=” **运算符，用于比较运算符两侧的操作数是否相等，如果两侧操作数相等返回值为 1，否则为 0。注意 NULL 不能用于“=”比较**

**“<>” 运算符，和“=”相反，如果两侧操作数不等，则值为 1，否则为 0。NULL 不能用于“<>”比较。**

**“<=>” 安全的等于运算符，和“=”类似，在操作数相等时值为 1，不同之处在于即使操作的值为 NULL 也可以正确比较**

**“<” 运算符，当左侧操作数小于右侧操作数时，其返回值为 1，否则其值为 0。**

**“<=” 运算符，当左侧操作数小于等于右侧操作数时，其返回值为 1，否则返回值为 0。**

**“>” 运算符，当左侧操作数大于右侧操作数时，其返回值为 1，否则返回值为 0。**

**“>=” 运算符，当左侧操作数大于等于右侧操作数时，其返回值为 1，否则返回值为 0。**

**“BETWEEN”运算符的使用格式为“a BETWEEN min AND max”，当 a 大于等于 min 并且小于等于 max，则返回值为 1，否则返回 0；当操作数 a、min、max 类型相同时，此表达式等价于（a>=min and a<=max），当操作数类型不同时，比较时会遵循类型转换原则进行转换后，再进行比较运算。**

**“IN” 运算符的使用格式为“a IN (value1,value2,…)”,当 a 的值存在于列表中时，则整个比较表达式返回的值为 1，否则返回 0。**

**“IS NULL”运算符的使用格式为“a IS NULL”,当 a 的值为 NULL，则返回值为 1，否则返回 0。**

**“IS NOT NULL”运算符的使用格式为“a IS NOT NULL”。和“IS NULL”相反，当 a 的值不为 NULL，则返回值为 1，否则返回 0。**

**“LIKE”运算符的使用格式为“a LIKE %123%”,当 a 中含有字符串“123”时，则返回值为 1，否则返回 0。**

**“REGEXP”运算符的使用格式为“str REGEXP str\_pat”,当 str 字符串中含有 str\_pat相匹配的字符串时，则返回值为 1，否则返回 0。（正则表达式匹配运算符）**

### （3）逻辑运算符

**逻辑运算符又称为布尔运算符，用来确认表达式的真和假。MySQL 支持 4 种逻辑运算符。**





**“NOT”或“！”表示逻辑非。返回和操作数相反的结果：当操作数为 0（假），则返回值为 1，否则值为 0。但是有一点除外，那就是 NOT NULL 的返回值为 NULL，这一点请大家注意。**

**“AND”或“&&”表示逻辑与运算。当所有操作数均为非零值并且不为 NULL 时，计算所得结果为 1，当一个或多个操作数为 0 时，所得结果为 0，操作数中有任何一个为NULL 则返回值为 NULL。**

**“OR”或“||”表示逻辑或运算。当两个操作数均为非 NULL 值时，如有任意一个操作数为非零值，则结果为 1，否则结果为 0。当有一个操作数为 NULL 时，如另一个操作数为非零值，则结果为 1，否则结果为 NULL。假如两个操作数均为 NULL，则所得结果为 NULL。**

**“XOR”表示逻辑异或。当任意一个操作数为 NULL 时，返回值为 NULL。对于非 NULL 的操作数，如果两个的逻辑真假值相异，则返回结果 1；否则返回 0。（即如果不为空的两个数，值不相等成立，相等不成立）**

### （4）位运算符

**位运算是将给定的操作数转化为二进制后，对各个操作数每一位都进行指定的逻辑运算，得到的二进制结果转换为十进制数后就是位运算的结果。MySQL 5.0 支持 6 种位运算符。（暂时了解）（是实话二进制这点不太明白）**



**可以发现，位运算符中的位与“&”和位或“|”和前面介绍的逻辑与和逻辑或非常类似。其他操作符和逻辑操作有所不同，**

**“位与”对多个操作数的二进制位作逻辑与操作，例如 2&3，因为 2 的二进制数是 10，3 是 11，所有 10&11 的结果是 10，十进制数字还是 2。**

**“位或”对多个操作数的二进制位作逻辑或操作，还是上面的例子，2|3 的结果应该是10|11，结果还是 11，应该是 3**

**“位异或”对操作数的二进制位做异或操作，10^11 的结果是 01，结果应该是 1。**

**“位取反”对操作数的二进制位作 NOT 操作，这里的操作数只能是一位，**

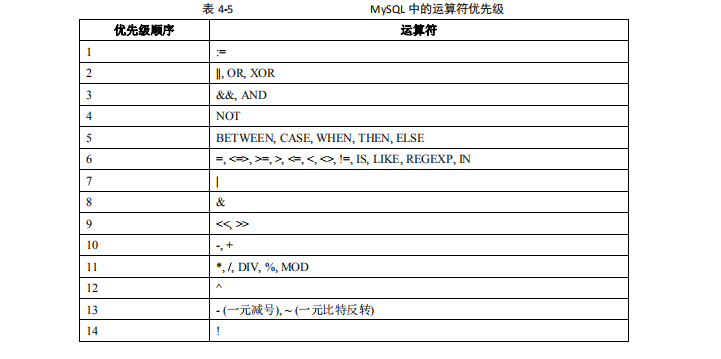
**结果让大家可能有些疑惑，1 的位取反怎么会是这么大的数字？来研究一下，在 MySQL 中，常量数字默认会以 8 个字节来表示，8 个字节就是 64 位，常量 1 的二进制表示为 63 个“0”加 1 个“1”，位取反后就是 63 个“1”加一个“0”，转换为二进制后就是 18446744073709551614，**

**“位右移”对左操作数向右移动右操作数指定的位数。例如 100>>3，就是对 100 的二进制数 0001100100 右移 3 位，左边补 0，结果是 0000001100，转换为二进制数是 12。**

**“位左移”对左操作数向左移动右操作数指定的位数。例如 100<<3，就是对 100 的二进制数 0001100100 左移 3 位，右边补 0，结果是 1100100000，转换为二进制数是 800，**

### （5）运算符的优先级

**表 4-5 中列出了所有的运算符，优先级由低到高排列，同一行中的运算符具有相同的优先级。**



**在实际运行的时候，可以参考表 4-5 中的优先级。实际上，很少有人能将这些优先级熟练记忆，很多情况下我们都是用“（）”来将需要优先的操作括起来，这样既起到了优先的作用，又使得其他用户看起来更加易于理解。**

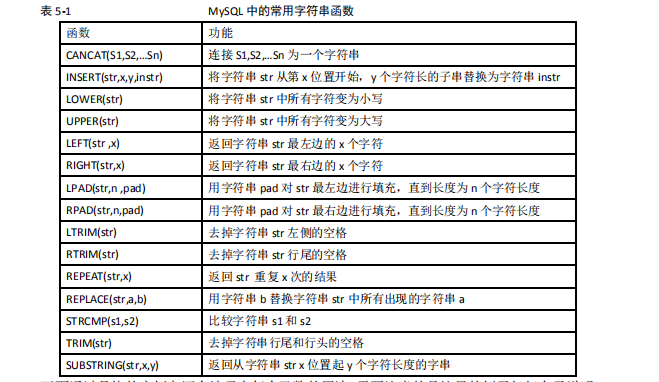
**这些运算符可以帮助用户完成算术、比较、逻辑和位逻辑操作，大家在使用时要注意运算符的优先级。另外，在使用比较运算符时要保证比较的操作数类型是一致的，这样可以避免由于操作数类型的不一致而得出错误的数据**

## 常用函数

**MySQL 提供了多种内建函数帮助开发人员编写简单快捷的 SQL 语句，其中常用的函数有字符串函数、日期函数和数值函数。**

### 字符串函数

字符串函数是最常用的一种函数了，如果大家编写过程序的话，不妨回过头去看看自己使用过的函数，可能会惊讶地发现字符串处理的相关函数占已使用过的函数很大一部分。MySQL中字符串函数也是最丰富的一类函数.



**CANCAT(S1,S2,…Sn)函数：把传入的参数连接成为一个字符串。下面的例子把“aaa”、“bbb”、“ccc”3 个字符串连接成了一个字符串“aaabbbccc”。另外，任何字符串与 NULL 进行连接的结果都将是 NULL。**

mysql> select concat('weiwei','fengyu','fengyu','wei'); #将字符串拼接至一起。

**INSERT(str ,x,y,instr)函数：将字符串 str 从第 x 位置开始，y 个字符长的子串替换为字符串 instr。**

mysql> select insert('weiwei123456',7,6,'yufeng'); #将制定的范围内的字符串替换

**LOWER(str)和 UPPER(str)函数：把字符串转换成小写或大写。在字符串比较中，通常要将比较的字符串全部转换为大写或者小写， （LOWER转小写，UPPER转大写）**

mysql> select lower('wei123456') , upper('wei123456'); #大小写转换

**LEFT(str,x)和 RIGHT(str,x)函数：分别返回字符串最左边的 x 个字符和最右边的 x 个字符。如果第二个参数是 NULL，那么将不返回任何字符串。（LEFT左边 RIGHT右边）**

mysql> select left('weiweia',3), right('weiweia',4), left(right('weiweia',4),1); #mysql 中的截取还挺方便

**LPAD(str,n ,pad)和 RPAD(str,n ,pad)函数：用字符串 pad 对 str 最左边和最右边进行填充,直到长度为 n 个字符长度。**

mysql> select lpad('weiwei',14,'tiantian'), rpad('weiwei',14,'tiantian'); #数据填充 pad

**LTRIM(str)和 RTRIM(str)函数：去掉字符串 str 左侧和右侧空格**

mysql> select ltrim(' --mysql') , rtrim('mysql-- '); #去除空格trim

**REPEAT(str,x)函数：返回 str 重复 x 次的结果。**

mysql> select repeat('mysql',3); #重复输出

**REPLACE(str,a,b)函数：用字符串 b 替换字符串 str 中所有出现的字符串 a。**

mysql> select replace('mysqljavascript','javascript','mysql'); #替换

**STRCMP(s1,s2)函数：比较字符串 s1 和 s2 的 ASCII 码值的大小。如果 s1 比 s2 小，那么返回-1；**

**如果 s1 与 s2 相等，那么返回 0；如果 s1 比 s2 大，那么返回 1。**

mysql> select strcmp(1,2) , strcmp(2,2) , strcmp(3,2); #用ASCLL码值做比较

**TRIM(str)函数：去掉目标字符串的开头和结尾的空格。**

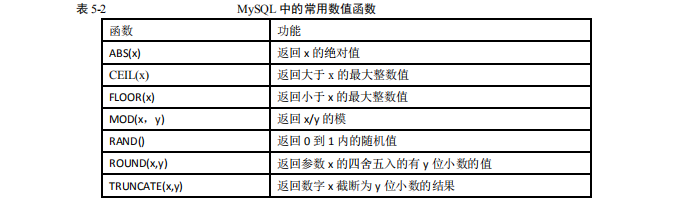
mysql> select trim(' --mysql-- '); #去吃空格

**SUBSTRING(str,x,y)函数：返回从字符串 str 中的第 x 位置起 y 个字符长度的字串。此函数经常用来对给定字符串进行字串的提取。**

mysql> select substring('mysql',3,3); # mysql中的字符串截取

### 数值函数

MySQL 中另外一类很重要的函数就是数值函数，这些函数能处理很多数值方面的运算。可以想象，如果没有这些函数的支持，用户在编写有关数值运算方面的代码时将会困难重重，举个例子，如果没有 ABS 函数的话，如果要取一个数值的绝对值，就需要进行好多次判断才能返回这个值，而数字函数能够大大提高用户的工作效率。



**ABS(x)函数：返回 x 的绝对值。**

mysql> select abs(-7) , abs(7);

**CEIL(x)函数：返回大于 x 的最大整数。**

mysql> select ceil(-6.8) , ceil(7.2); #向上取整

**FLOOR(x)函数：返回小于 x 的最大整数，和 CEIL 的用法刚好相反。**

mysql> select floor(-6.8) , floor(7.2); #向下取整

**MOD(x，y)函数：返回 x/y 的模。和 x%y 的结果相同，模数和被模数任何一个为 NULL 结果都为 NULL。**

mysql> select mod(3,2) , mod(15,7); #整除取余

**RAND()函数：返回 0 到 1 内的随机值。每次执行结果都不一样。**

mysql> select rand() , rand(); #随机数，小数点后一共十六位数字

**ROUND(x,y)函数：返回参数 x 的四舍五入的有 y 位小数的值。如果是整数，将会保留 y 位数量的 0；如果不写 y，则默认 y 为 0，即将 x 四舍五入后取整。适合于将所有数字保留同样小数位的情况。**

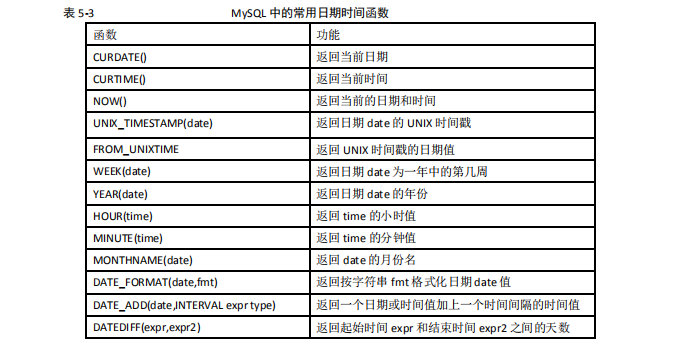
mysql> select round(1.1) , round(1.1,2) , round(1,2); #四舍五入保留小数，后面得值代表保留几位小数

**TRUNCATE(x,y)函数：返回数字 x 截断为 y 位小数的结果。注意 TRUNCATE 和 ROUND 的区别在于 TRUNCATE 仅仅是截断，而不进行四舍五入。**

mysql> select truncate(1.12356,2); #直接截断小数点Y位以后的值

### （3）日期和时间函数

有时我们可能会遇到这样的需求：当前时间是多少、下个月的今天是星期几、统计截止到当前日期前 3 天的收入总和等。这些需求就需要日期和时间函数来实现。



**CURDATE()函数：返回当前日期，只包含年月日。**

mysql> select curdate();

**CURTIME()函数：返回当前时间，只包含时分秒。**

mysql> select curtime();

**NOW()函数：返回当前的日期和时间，年月日时分秒全都包含。**

mysql> select now();

**UNIX\_TIMESTAMP(date)函数：返回日期 date 的 UNIX 时间戳。**

mysql> select unix\_timestamp(); #默认值为当前时间

**FROM\_UNIXTIME （ unixtime ） 函 数 ： 返 回 UNIXTIME 时 间 戳 的 日 期 值 ， 和UNIX\_TIMESTAMP(date)互为逆操作。**

mysql> select from\_unixtime('1550303602'); #时间戳转换为当前时间，这个没有默认值，必须添值

**WEEK(DATE)和 YEAR(DATE)函数：前者返回所给的日期是一年中的第几周，后者返回所给的日期是哪一年。**

mysql> select week(now()) , year(now()); #必须有默认的值。

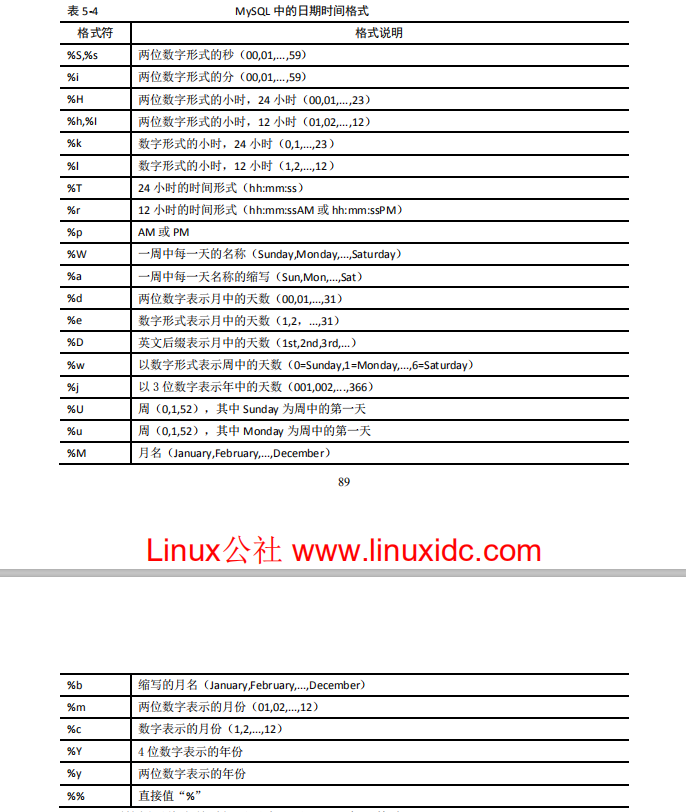
**HOUR(time)和 MINUTE(time)函数：前者返回所给时间的小时，后者返回所给时间的分钟。**

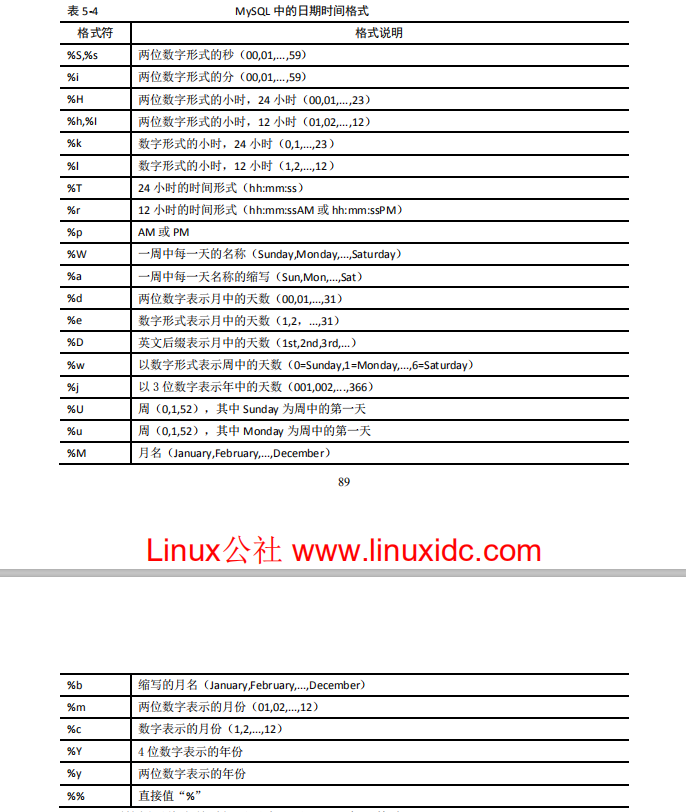
mysql> select hour(curtime()) , minute(curtime()); #给定的时间必须有小时和分钟

**MONTHNAME(date)函数：返回 date 的英文月份名称。**

mysql> select monthname(curdate());

**DATE\_FORMAT(date,fmt)函数：按字符串 fmt 格式化日期 date 值，此函数能够按指定的格式显示日期，**

****



**DATE\_ADD(date,INTERVAL expr type)函数：返回与所给日期 date 相差 INTERVAL 时间段的日期。**

**其中 INTERVAL 是间隔类型关键字，expr 是一个表达式，这个表达式对应后面的类型，type是间隔类型，MySQL 提供了 13 种间隔类型。**



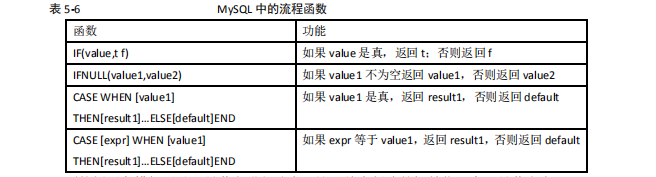
mysql> select now() current , date\_add(now(),INTERVAL 31 day);

**DATEDIFF（date1，date2）函数：用来计算两个日期之间相差的天数。**

mysql> select datediff('20190501',curdate());

### （4）流程函数

流程函数也是很常用的一类函数，用户可以使用这类函数在一个 SQL 语句中实现条件选择，这样做能够提高语句的效率。



**IF(value,t,f)函数：**

mysql> select if (salary>2000,'high','low') from salary; # 如果salary字段大于2000则返回high 否则返回low

**IFNULL(value1,value2)函数：这个函数一般用来替换 NULL 值的，我们知道 NULL 值是不能参与数值运算的.**

mysql> select ifnull(salary,0) from salary;  **#如果salary值为null 则用0替换它**

**CASE WHEN [value1] THEN[result1]…ELSE[default]END 函 数 ：**

mysql> select case when salary>2000 then 'high' else 'low' end from salary;

**#SQL中的if()else 如果salary字段的值大于2000则输出hign否则输出low**

**CASE [expr] WHEN [value1] THEN[result1]…ELSE[default]END 函数：**

mysql> select case salary when 1000 then '1' when 2000 then '2' when 3000 then '3' else '4' end from salary; **#SQL中的ifelse if格式**

### （5）其他常用函数



**DATABASE()函数：返回当前数据库**

mysql> select database();

**VERSION()函数：返回当前数据库版**

mysql> select version();

**USER()函数：返回当前登录用户名。**

mysql> select user();

**INET\_ATON(IP)函数：返回 IP 地址的网络字节序表示。**

mysql> select INET\_ATON('192.168.1.1');

**INETNTOA(num)函数：返回网络字节序代表的 IP 地址。**

mysql> select inet\_ntoa(3232235777);

**INET\_ATON(IP)和 INET\_NTOA(num)函数主要的用途是将字符串的 IP 地址转换为数字表示的网**

**络字节序，这样可以更方便地进行 IP 或者网段的比较。**

mysql> select \* from t where inet\_aton(ip) >= inet\_aton('192.168.1.3') and inet\_aton(ip) <= inet\_aton('192.168.1.20');

**PASSWORD(str)函数：返回字符串 str 的加密版本，一个 41 位长的字符串。**

**此函数只用来设置系统用户的密码，但是不能用来对应用的数据加密。如果应用方面有加密的需求，可以使用 MD5 等加密函数来实现。**

mysql> select password('123456');

**MD5(str)函数：返回字符串 str 的 MD5 值，常用来对应用中的数据进行加密。**

mysql> select md5('123456');

**本章主要对 MySQL 常用的各类常用函数通过实例做了介绍。MySQL 有很多内建函数，这些内建函数实现了很多应用需要的功能并且拥有很好的性能，如果用户在工作中需要实现某种功能，最好先查一下 MySQL 官方文档或者帮助，看是否已经有相应的函数实现了我们需要的功能，可以大大提高工作效率。**

# 开发篇

## 存储引擎

### 存储引擎概述

插件式存储引擎是 MySQL 数据库最重要的特性之一，用户可以根据应用的需要选择如何存储和索引数据、是否使用事务等。MySQL 默认支持多种存储引擎，以适用于不同领域的数据库应用需要，用户可以通过选择使用不同的存储引擎提高应用的效率，提供灵活的存储，用户甚至可以按照自己的需要定制和使用自己的存储引擎，以实现最大程度的可定制性

**MySQL 5.0 支持的存储引擎包括 MyISAM、InnoDB、BDB、MEMORY、MERGE、EXAMPLE、NDB Cluster、ARCHIVE、CSV、BLACKHOLE、FEDERATED 等，其中 InnoDB 和 BDB 提供事务安全表，其他存储引擎都是非事务安全表。**

**默认情况下，创建新表不指定表的存储引擎，则新表是默认存储引擎的，如果需要修改默认的存储引擎，则可以在参数文件中设置 default-table-type。**

**查看当前的默认存储引擎下命令： show variables like 'tabletype'; （‘tabletype’在5.53版本后已取消，了解即可）**

**两种方法查询当前数据库支持的存储引擎:**

**①mysql> SHOW ENGINES \G;**

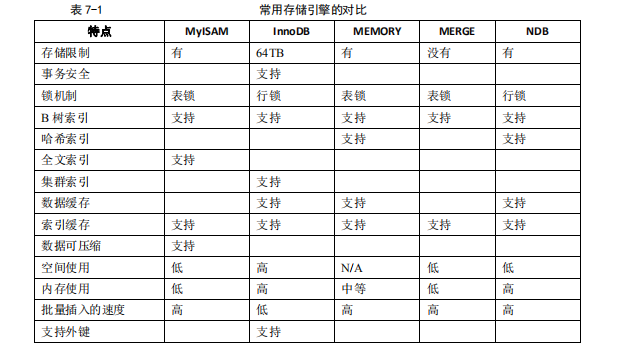
**②mysql> SHOW VARIABLES LIKE 'have%';**

**其中 Value 显示为“DISABLED”的记录表示支持该存储引擎，但是数据库启动的时候被禁用。**

**存储引擎修改：mysql> alter table ai engine = innodb;**

### 存储引擎特性

下面重点介绍几种常用的存储引擎,并对比各个存储引擎之间的区别，以帮助读者理解不同存储引擎的使用方式



**下面将重点介绍最常使用的 4 种存储引擎：MyISAM、InnoDB、MEMORY 和 MERGE。NDB存储引擎会在第 33 章 MySQL CLUSTER 中详细介绍，这里不再赘述。**

### MyISAM引擎

**MyISAM 是 MySQL 的默认存储引擎。MyISAM 不支持事务、也不支持外键，其优势是访问的速度快，对事务完整性没有要求或者以 SELECT、INSERT 为主的应用基本上都可以使用这个引擎来创建表。**

**每个 MyISAM 在磁盘上存储成 3 个文件，其文件名都和表名相同，但扩展名分别是：**

** .frm（存储表定义）；**

** .MYD（MYData，存储数据）；**

** .MYI （MYIndex，存储索引）。**

**数据文件和索引文件可以放置在不同的目录，平均分布 IO，获得更快的速度**

**要指定索引文件和数据文件的路径，需要在创建表的时候通过 DATA DIRECTORY 和 INDEX DIRECTORY 语句指定，也就是说不同 MyISAM 表的索引文件和数据文件可以放置到不同的路径下。文件路径需要是绝对路径，并且具有访问权限。**

**MyISAM 类型的表可能会损坏，原因可能是多种多样的，损坏后的表可能不能访问，会提示需要修复或者访问后返回错误的结果。MyISAM 类型的表提供修复的工具，可以用 CHECK TABLE 语句来检查 MyISAM 表的健康，并用 REPAIR TABLE 语句修复一个损坏的 MyISAM 表。表损坏可能导致数据库异常重新启动，需要尽快修复并尽可能地确认损坏的原因。具体的操作步骤可以参见第 35 章应急处理。**

**MyISAM 的表又支持 3 种不同的存储格式，分别是：**

** 静态（固定长度）表；**

**静态表是默认的存储格式。静态表中的字段都是非变长字段，这样每个记录都是固定长度的。**

**优点：存储非常迅速，容易缓存，出现故障容易恢复；**

**缺点：占用的空间通常比动态表多。**

**静态表的数据在存储的时候会按照列的宽度定义补足空格，但是在应用访问的时候并不会得到这些空格，这些空格在返回给应用之前已经去掉。**

**注意：但是也有些需要特别注意的问题，如果需要保存的内容后面本来就带有空格，那么在返**

**回结果的时候也会被去掉，开发人员在编写程序的时候需要特别注意，因为静态表是默认的存储格式，开发人员可能并没有意识到这一点，从而丢失了尾部的空格。**

** 动态表；**

**动态表中包含变长字段，记录不是固定长度的。**

**优点：占用的空间相对较少，**

**缺点：频繁地更新删除记录会产生碎片，需要定期执行 OPTIMIZE TABLE 语句或 myisamchk -r 命**

**令来改善性能，并且出现故障的时候恢复相对比较困难。**

** 压缩表。**

**压缩表由 myisampack 工具创建，占据非常小的磁盘空间。因为每个记录是被单独压缩的，所以只有非常小的访问开支。**

### InnoDB引擎

**InnoDB 存储引擎提供了具有提交、回滚和崩溃恢复能力的事务安全。但是对比 MyISAM的存储引擎，InnoDB 写的处理效率差一些并且会占用更多的磁盘空间以保留数据和索引。**

**InnoDB 存储引擎的表使用过程中不同于其他存储引擎的特点。**

**①自动增长列**

**InnoDB 表的自动增长列可以手工插入，但是插入的值如果是空或者 0，则实际插入的将是自动增长后的值。**

mysql> create table autoincre\_demo (i smallint not null auto\_increment , name varchar(10) , primary key(i))engine=innodb;

mysql> insert into autoincre\_demo values (1,'1') , (0,'2') , (null,'3');

mysql> select \* from autoincre\_demo;

**可以通过“ALTER TABLE \*\*\* AUTO\_INCREMENT = n;”语句强制设置自动增长列的初识值，默认从 1 开始，但是该强制的默认值是保留在内存中的，如果该值在使用之前数据库重新启动，那么这个强制的默认值就会丢失，就需要在数据库启动以后重新设置**

**可以使用 LAST\_INSERT\_ID()查询当前线程最后插入记录使用的值。如果一次插入了多条记录，那么返回的是第一条记录使用的自动增长值。**

mysql> insert into autoincre\_demo values (4,'4');

mysql> select LAST\_INSERT\_ID();

mysql> insert into autoincre\_demo(name) values ('5'), ('6'), ('7');

mysql> select LAST\_INSERT\_ID();

**对于 InnoDB 表，自动增长列必须是索引。如果是组合索引，也必须是组合索引的第一列，但是对于 MyISAM 表，自动增长列可以是组合索引的其他列，这样插入记录后，自动增长列是按照组合索引的前面几列进行排序后递增的。（不太懂这点）**

mysql> create table autoincre\_demo (d1 smallint not null auto\_increment , d2 smallint not null , name varchar(10) , index(d2,d1) ) engine=myisam;

mysql> insert into autoincre\_demo (d2,name) values (2,'2'),(3,'3'),(4,'4'),(2,'2'),(3,'3'),(4,'4');

**②外键约束**

**MySQL 支持外键的存储引擎只有 InnoDB，在创建外键的时候，要求父表必须有对应的索引，子表在创建外键的时候也会自动创建对应的索引**

mysql> create table country(country\_id smallint unsigned not null auto\_increment,country varchar(50) not null ,last\_update timestamp not null default current\_timestamp on update current\_timestamp, primary key(country\_id) )engine=innodb default charset=utf8;

mysql> create table city(city\_id smallint unsigned not null auto\_increment,cityvarchar(50) not null , country\_id smallint unsigned not null, last\_update timestamp not null default current\_timestamp on update current\_timestamp,primary key(city\_id), key idx\_fk\_country\_id(country\_id),constraint fk\_city\_country foreign key (country\_id) references country(country\_id) on delete restrict on update cascade)engine=innodb default charset=utf8;

**在创建索引的时候，可以指定在删除、更新父表时，对子表进行的相应操作，包 RESTRICT、CASCADE、SET NULL 和 NO ACTION。**

**其中 RESTRICT 和 NO ACTION 相同，是指限制在子表有关联记录的情况下父表不能更新；**

**CASCADE 表示父表在更新或者删除时，更新或者删除子表对应记录；**

**SET NULL 则表示父表在更新或者删除的时候，子表的对应字段被 SET NULL。**

**选择后两种方式的时候要谨慎，可能会因为错误的操作导致数据的丢失。**

mysql> insert into country (country\_id,country,last\_update) values(1,'Afghanistn',now());

mysql> insert into city (city\_id,city,country\_id,last\_update) values(251,'Kabul',1,now());

mysql> update country set country\_id=1000 where country\_id=1;

**上面两个表，子表的外键指定是 ON DELETE RESTRICT ON UPDATE CASCADE方式的，那么在主表删除记录的时候，如果子表有对应记录，则不允许删除，主表在更新记录的时候，如果子表有对应记录，则子表对应更新**

**当某个表被其他表创建了外键参照，那么该表的对应索引或者主键禁止被删除。**

**在导入多个表的数据时，如果需要忽略表之前的导入顺序，可以暂时关闭外键的检查；**

**同样，在执行 LOAD DATA 和 ALTER TABLE 操作的时候，可以通过暂时关闭外键约束来加快处理的速度，关闭的命令是“SET FOREIGN\_KEY\_CHECKS = 0;”，执行完成之后，通过执行“SET FOREIGN\_KEY\_CHECKS = 1;”语句改回原状态。**

**对于 InnoDB 类型的表，外键的信息通过使用 show create table 或者 show table status 命令都可以显示。**

**③存储方式**

**InnoDB 存储表和索引有以下两种方式。**

**使用共享表空间存储，这种方式创建的表的表结构保存在.frm 文件中，数据和索引保存在 innodb\_data\_home\_dir 和 innodb\_data\_file\_path 定义的表空间中，可以是多个文件。**

**使用多表空间存储，这种方式创建的表的表结构仍然保存在.frm 文件中，但是每个表的数据和索引单独保存在.ibd 中。如果是个分区表，则每个分区对应单独的.ibd文件，文件名是“表名+分区名”，可以在创建分区的时候指定每个分区的数据文件的位置，以此来将表的 IO 均匀分布在多个磁盘上。**

**要使用多表空间的存储方式，需要设置参数 innodb\_file\_per\_table，并重新启动服务后才可以生效，对于新建的表按照多表空间的方式创建，已有的表仍然使用共享表空间存储。**

**如果将已有的多表空间方式修改回共享表空间的方式，则新建表会在共享表空间中创建，但已有的多表空间的表仍然保存原来的访问方式。所以多表空间的参数生效后，只对新建的表生效。**

**多表空间的数据文件没有大小限制，不需要设置初始大小，也不需要设置文件的最大限制、扩展大小等参数。**

**对于使用多表空间特性的表，可以比较方便地进行单表备份和恢复操作，但是直接复制.ibd 文件是不行的，因为没有共享表空间的数据字典信息，直接复制的.ibd 文件和.frm 文件恢复时是不能被正确识别的，但可以通过以下命令：**

**ALTER TABLE tbl\_name DISCARD TABLESPACE;**

**ALTER TABLE tblname IMPORT TABLESPACE**

**将备份恢复到数据库中，但是这样的单表备份，只能恢复到表原来在的数据库中，而不能恢复到其他的数据库中。如果要将单表恢复到目标数据库，则需要通过 mysqldump 和mysqlimport 来实现。**

### （5）MEMOY引擎

### （5）选择合适的引擎

### 