MySQL是一个数据库管理系统，也是一个关系数据库。它是由Oracle支持的开源软件。MySQL的官方网址： <http://www.mysql.com/> ，MySQL的社区版本下载地址为： <http://dev.mysql.com/downloads/mysql/> ；当前主流的关系型数据库有Oracle、DB2、Microsoft SQL Server、Microsoft Access、MySQL、浪潮K-DB等

数据类型：

MySQL使用许多不同的数据类型分为三类：数值型（整型，浮点型），日期/时间和文本类型，复合型；

**插入中文乱码问题：**

* 显示当前编码状态：

SHOW VARIABLES LIKE “%char%”;

* 设置显示编码为GBK

SET NAMES gbk;

Ps:并未改变实际编码，只是设置显示编码格式；这种方式笨重启无效，可以直接修改my.ini属性文件default-character-set=gbk；只有在windows时才会修改；

另外win10 cmd窗口需要使用旧版，新版修改属性无效；

**数据库的链接与退出：**

1. 连接数据库服务：

Mysql -u‘root’ [-h‘localhost:3306’] -p //Please enter password:

1. 断开数据库服务：

QUIT / EXIT

[http://www.runoob.com/sql/sql-tutorial.html //](http://www.runoob.com/sql/sql-tutorial.html%20//) 参考

## **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 ;只处理表及库，不对表中数据进行处理；

* 创建数据库：

CREATE DATABASE [IF NO EXISTS] db\_name [DEFAULT CHARSET utf8]；

* 显示当前所有数据库列表：

SHOW DATABASES; //5 rows in set

* 删除数据库：

DROP DATABASE [IF EXISTS ] db\_name； //Query ok,0 row affected

* 使用/切换到指定的数据库：

USE db\_name； //Database changed

* 查看当前使用的数据库及表：

SELECT DATABASE(); //如果没有use database则返回null；

* 查看当前库连接用户：

SELECT USER();

* 查看数据库版本;

SELECT VERSION()；

* 导入其他库中数据表：

SOURCE ‘D:/\*.sql’;

* 创建数据表：

CREATE TABLE [IF NOT EXISTS] t\_name (

column\_name1 column\_type column\_modifier1 column\_modifier2 column\_modifier3 ‘…’, //每个属性以空格隔开，每个column以，隔开

column\_name2 column\_type column\_modifier1 column\_modifier2 column\_modifier3 ‘…’,

. . .)；

PS: 字段修饰：

-1 <PRIMARY KEY> 主键；

-2 <AUTO\_INCREMENT> 自增，修饰整形数据字段，通常与主键配合使用；

-3 <UNSIGNED> 无正负符号，通常修饰数值型字段；更多修饰TINYINT类型

-4 <DEFAULT> 设置字段默认值，如果语句没有声明具体值将赋予对应数据该字段的默认值；

-5 <NOT NULL | NULL> 字段数据是否允许值为null；

-6 <UNIQUE(唯一索引) INDEX(普通索引) FULLTEXT(全文索引)>增加索引；

-7 <COMMENT ‘msg’> 增加字段说明；

Ps:使用或创建数据表前要USE对应的数据库；

* 显示当前数据库中所有的表；

SHOW TABLES;

* 查看制定表的結構；

DESC /DESCRIBE t\_name;

SHOW COLUMNS FROM t\_name；

* 删除数据表文件本身：

DROP TABLE t\_name;

* 查看创建表的语句：

SHOW CREATE TABLE t\_name;

----------------------------------------------------------------

ALTER TABLE 语句用于在已有的表中添加、删除或修改列

* 向已有数据表中增加字段：(可以指定在哪个字段后面)

ALTER TABLE t\_name ADD col\_name col\_type… [AFTER exist\_col\_name];

* 修改列字段的属性：

ALTER TABLE t\_name MODIFY col\_name type\_value;

* AlTER TABLE t\_name CHANGE col\_name new\_col\_name typevalue;

Ps: MODIFY 不能修改字段名，适用于修改表字段属性，CHANGE可以；

* 删除表中已有字段及对应字段的数据：

ALTER TABLE t\_name DROP col\_name;

* 修改/重命名数据表名：

ALTER TABLE t\_name RENAME [TO] new\_t\_name;

* 取消表主键：

ALTER TABLE t\_name DROP PRIMARY KEY;

//ALTER 修改已有表，只有ADD,CHANGE.MODIFY可以使用AFTER | FIREST;

注意：AFTER，FIREST要在所有属性的最后面声明；

----------------------------------------------------------------

Example: 创建一个表；

mysql> CREATE TABLE t\_employee(

-> id INT(8) PRIMARY KEY AUTO\_INCREMENT COMMENT '員工ID',

-> name VARCHAR(50) NOT NULL COMMENT ‘員工个人姓名’,

-> gender CHAR DEFAULT ‘0’ COMMENT ‘員工性別,0男1女,默认0’,

-> age TINYINT(3) UNSIGHED DEFAULT 18 COMMENT ‘員工个人年龄’,

-> position VARCHAR(30) COMMENT ‘員工的工作職位’,

-> salary FLOAT(6,2) DEFAULT 0.00 COMMENT ‘員工的月薪’

-> phone\_number VARCHAR(11) NOT NULL COMMENT ‘員工個人移動電話號碼’,

-> home\_address VARCHAR(60) NOT NULL COMMENT ‘員工的家庭住址’,

);

修改已有的属性；

Mysql> ALTER TABLE t\_employee MODIFY

-> GENDER TINYINT

-> DEFAULT 2

-> COMMENT ‘0表示女性，1表示男性，2表示未知’；

增加一个字段在age后面：

Mysql> ALTER TABLE t\_employee

-> ADD COLUMN per\_info varchar(30)

-> AFTER age

-> comment ‘间的信息介绍’;

**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。(增添改查）

//只处理表中的数据，不对表本身处理;

* 插入数据到指定的数据表中：

1. INSERT INTO t\_name (field1, field2,...fieldN )

VALUES

( value1, value2,...valueN );

1. INSERT INTO t\_name VALUES

(value1, value2,...valueN );

Ps:可以不加字段，但需要与字段值的全顺序完全相同完整的values，包括自增的ID也要赋值；如果是第一条数据给出的自增ID有效，其他无效；第一列如果没有设置主键自增（PRINARY KEY AUTO\_INCREMENT）的话添加第一列数据比较容易错乱，要不断的查询表看数据。

如果添加过主键自增（PRINARY KEY AUTO\_INCREMENT）第一列在增加数据的时候，可以写为0或者null，这样添加数据可以自增， 从而可以添加全部数据，而不用特意规定那几列添加数据

* 更新表格数据：

UPDATE t\_name SET field1=new-value1, field2=new-value2…

[WHERE Clause]；

或改多张表：把多张表声明为简写，如t1，t2…；

UPDATE t1\_name t1，t2\_name t2 ,……

SET

t1.col\_name1 = new\_name1, t1.col\_name2 = new\_name2,…

t2.col\_name1 = new\_na1,t2.col\_name2=new\_na2,…

… [WHERE Clause]；

* 删除表格数据：

1. DELETE FROM t\_name [WHERE Clause]；

Ps：如果没有条件会删除整个表的数据；id会不变继续累加；

1. TRUNCATE t\_name;

Ps:同样是删除数据，这个方法是清空数据；id也会重置；

1、当你不再需要该表时， 用 **drop**;

2、当你仍要保留该表，但要删除所有记录时， 用 **truncate**;

3、当你要删除部分记录时， 用 **delete**。

**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。

在原有四个数据库中有个mysql库，里面包含数据库的数据基础信息；user表是用户表；

* 查询数据库用户：

USE mysql;

SELECT user,host,password FROM user;

* 从修改数据库密码：

1. UPDATE user SET password = PASSWORD(“value”) WHERE user = “user”;

Exmple: UPDATE user set password = PASSWORD(“123456”) WHERE user = “root”;

1. Mysqladmin -uroot -padmin password 123456;

* 数据安全连接方案：

1. 建议不使用外部ip进行连接，即不建议打开远程连接；
2. 使用本地localhost连接；
3. 设置其他可访问数据账户；

UPDATE user SET host= “127.0.0.1” WHERE user = “root”;

复合函数：

Max();

Min();

Sum()；

Count();

Avg();

**DQL（Data Query Languages）语句：**数据库查询语言；

查询数据：

SELECT

column\_1, column\_2, ...

FROM

table\_1

[INNER | LEFT | RIGHT] JOIN table\_2 ON conditions

WHERE

conditions

GROUP BY exist\_column

HAVING group\_conditions

ORDER BY exist\_column [ASC | DESC]

LIMIT [ offset ], length;

SELECT语句由以下列表中所述的几个子句组成：

* SELECT之后是逗号分隔列或星号(\*)的列表，表示要返回所有列。
* FROM指定要查询数据的表或视图。
* JOIN根据某些连接条件从其他表中获取数据。
* WHERE过滤结果集中的行。
* GROUP BY将一组行组合成小分组，并对每个小分组应用聚合函数。
* HAVING过滤器基于GROUP BY子句定义的小分组。
* ORDER BY指定用于排序的列的列表，使用参数DESC | ASC(降序|升序)。
* LIMIT限制返回行的数量(单数是返回数据从0开始的数量如:limit 5，双数返回数据所在是位置开始的数量如:limit 2,5，表示从2开始算后5个数据)。
* 可以使用分隔的一个或多个逗号从多个表，以及使用WHERE子句包括各种条件，但WHERE子句是SELECT命令的可选部分
* 可以在一个SELECT命令指定读取一个或多个字段
* 可以指定星号(\*)代替选择的字段。在这种情况下，将返回所有字段
* 可以指定任意的条件在 WHERE 子句后面
* 可以使用OFFSET指定一个偏移量，SELECT从那里开始返回记录。默认情况下 offset 的值是 0

注意：**BINARY** 区分所查条件与结果字符之间大小写；SQL-> LIKE 子句中使用百分号 **%**字符来表示任意字符，类似于UNIX或正则表达式中的星号 **\***

LIKE :

过滤指定字段含有某些字符的查询结果；“相似过滤”

1. 子句中使用百分号 **%**字符来表示任意字符，类似于UNIX或正则表达式中的星号 **\***。

2. 如果没有使用百分号 **%**, LIKE 子句与等号 **=** 的效果是一样的。

SELECT \* from t\_anme WHERE col\_name LIKE '%com';

'%a' //以a结尾的数据

'a%' //以a开头的数据

'%a%' //含有a的数据

'\_a\_' //三位且中间字母是a的

'\_a' //两位且结尾字母是a的

'a\_' //两位且开头字母是a的

ORDER BY:

把搜索结果排序：ASC（升序） / DESC（降序）

SELECT \* from t\_name ORDER BY col\_name ASC|DESC;

GROUP BY:

按照字段进行分组，通常配合AVG(),SUM()等函数使用；

SELECT name,count(\*) AS num from t\_name GROUP BY col\_name ORDER BY COUNT(\*) ASC;

以name为单位返回不重复name及name数量数据；

* 简要查询：

SELECT \* FROM t\_name;

* 标准表达式：

SELECT field1, field2,...fieldN FROM

->table\_name1, table\_name2...

->[ WHERE condition1 [AND [OR]] condition2.....

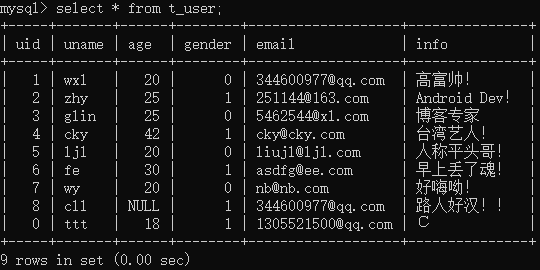
* 带有过滤查询（过滤重复数据）：

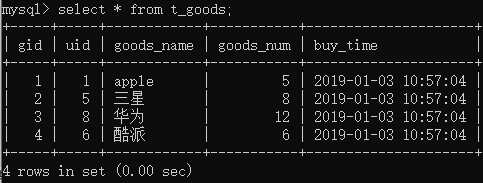
SELECT DISTINCT column FROM t\_name [ WHERE clause ];

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **示例** |
| = | 检查两个操作数的值是否相等 | (A = B) 不为 true. |
| != | 检查两个操作数的值是否相等 | (A != B) 为 true. |
| > | 检查左操作数的值是否大于右操作数的值 | (A > B) 不为 true. |
| < | 检查左操作数的值是否小于右操作数的值 | (A < B) 为 true. |
| >= | 检查左操作数的值是否大于或等于右操作数的值 | (A >= B) 不为 true. |
| <= | 检查左操作数的值是否小于或等于右操作数的值 | (A <= B) 为 true. |
| OR ,|| | 或 |  |
| And ,&& | 与 |  |
| Between..and | 在什么什么之间 |  |
| In | 在 .. 之内 |  |
| Not in | 不在 …之内 |  |

**Join联查：多张表一同查；**

例子：现有两张表：t\_user,t\_goods:





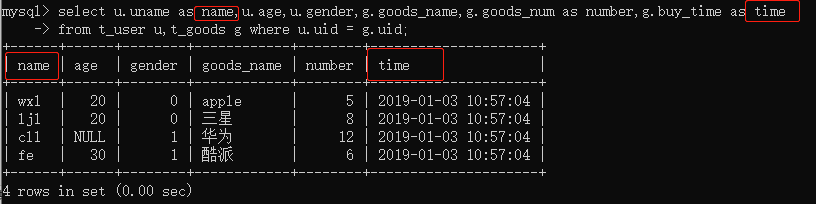
* 基本语法： //每个字段都可以声明别名，对位获取数据后的参数；
  1. SELECT t1.col\_name1 [ as T ],t1.col\_name2 [ as T2 ],...

->t2.col\_name1 [ as T3 ],t2.col\_name2 [ as T4 ],...

->FROM t1\_name t1,t2\_name t2,...

[ WHERE cluese ]

Ps: 这样可以简化表名；减少出错；

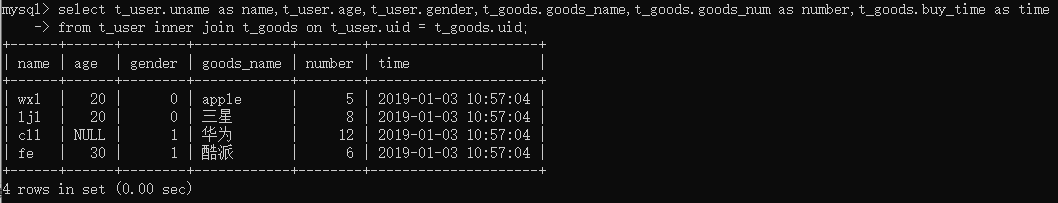


* 1. SELECT t1.col\_name1,t1.col\_name2 ,...

->t2.col\_name1,t2.col\_name2 ,...

->FROM t1\_name t1 INNER JOIN t2\_name t2,...

-> ON [ cluase ]





外联联查分为LEFT 和 RIGHT;

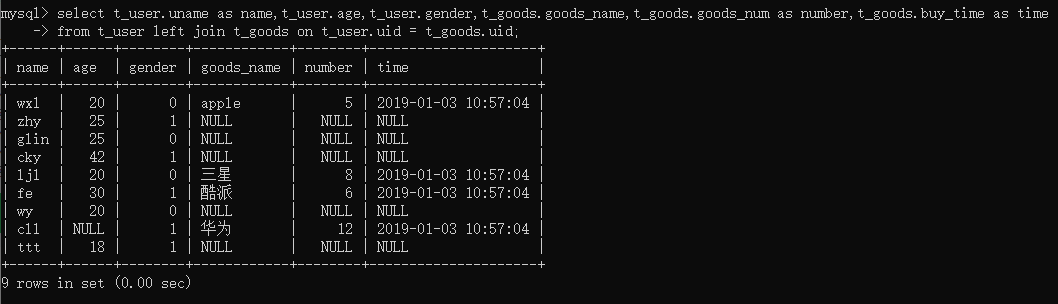
* **左连接：包含所有的左边表中的记录甚至是右边表中没有和它匹配的记录**

SELECT t1.col\_name1,t1.col\_name2 ,...

->t2.col\_name1,t2.col\_name2 ,...

->FROM t1\_name t1 LEFT JOIN t2\_name t2,...

-> ON [ cluase ]





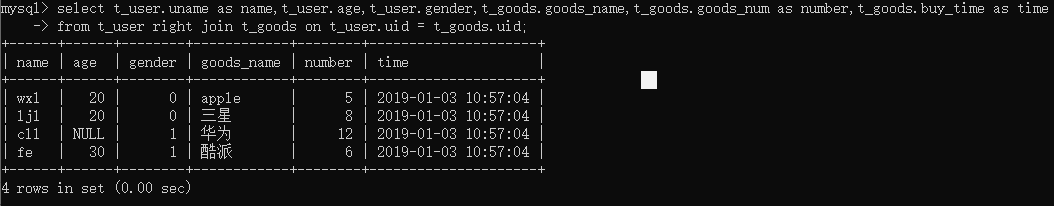
* **右连接：包含所有的右边表中的记录甚至是右边表中没有和它匹配的记录**

SELECT t1 name.col\_name1,t1\_name.col\_name2 ,...

->t2\_name.col\_name1,t2\_name.col\_name2 ,...

->FROM t1\_name RIGHT JOIN t2\_name,...

-> ON [ cluase ]





* **子查询：包含关键字in、not in、=、!=、exists、not exists等。**

SELECT col\_name1,col\_name2,...

->FROM t\_name

->WHERE uid

->IN (SELECT uid FROM T2\_NAME);



* **IS NULL:** 当列的值是 NULL,此运算符返回 true。
* **IS NOT NULL:** 当列的值不为 NULL, 运算符返回 true。
* **<=>:** 比较操作符（不同于=运算符），当比较的的两个值为 NULL 时返回 true。

SELECT \* FROM t\_name WHERE col\_name IS NULL;

## 数据类型：

MySQL 5.0 以上的版本：MySQL支持多种类型，大致可以分为三类：数值、日期/时间和字符串(字符)类型

1、一个汉字占多少长度与编码有关：

**UTF－8**：一个汉字＝3个字节

**GBK**：一个汉字＝2个字节

2、varchar(n) 表示 n 个字符，无论汉字和英文，Mysql 都能存入 n 个字符，仅是实际字节长度有所区别

3、MySQL 检查长度，可用 SQL 语言来查看：

select LENGTH(fieldname) from tablename

**整型**

|  |  |
| --- | --- |
| **MySQL数据类型** | **含义（有符号）** |
| tinyint(m) | 1个字节  范围(-128~127) |
| smallint(m) | 2个字节  范围(-32768~32767) |
| mediumint(m) | 3个字节  范围(-8388608~8388607) |
| int(m) | 4个字节  范围(-2147483648~2147483647) |
| bigint(m) | 8个字节  范围(+-9.22\*10的18次方) |

取值范围如果加了 unsigned，则最大值翻倍，如 tinyint unsigned 的取值范围为(0~256)。

int(m) 里的 m 是表示 SELECT 查询结果集中的显示宽度，并不影响实际的取值范围，没有影响到显示的宽度，不知道这个 m 有什么用。

**2、浮点型(float 和 double)**

|  |  |
| --- | --- |
| **MySQL数据类型** | **含义** |
| float(m,d) | 单精度浮点型    8位精度(4字节)     m总个数，d小数位 |
| double(m,d) | 双精度浮点型    16位精度(8字节)    m总个数，d小数位 |

设一个字段定义为 float(5,3)，如果插入一个数 123.45678,实际数据库里存的是 123.457，但总个数还以实际为准，即 6 位。

**3、定点数**

浮点型在数据库中存放的是近似值，而定点类型在数据库中存放的是精确值。

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

**4、字符串(char,varchar,\_text)**

|  |  |
| --- | --- |
| **MySQL数据类型** | **含义** |
| char(n) | 固定长度，最多255个字符 |
| varchar(n) | 可变长度，最多65535个字符 |
| tinytext | 可变长度，最多255个字符 |
| text | 可变长度，最多65535个字符 |
| mediumtext | 可变长度，最多2的24次方-1个字符 |
| longtext | 可变长度，最多2的32次方-1个字符 |

char 和 varchar：

* 1.char(n) 若存入字符数小于n，则以空格补于其后，查询之时再将空格去掉。所以 char 类型存储的字符串末尾不能有空格，varchar 不限于此。
* 2.char(n) 固定长度，char(4) 不管是存入几个字符，都将占用 4 个字节，varchar 是存入的实际字符数 +1 个字节（n<=255）或2个字节(n>255)，所以 varchar(4),存入 3 个字符将占用 4 个字节。
* 3.char 类型的字符串检索速度要比 varchar 类型的快。

varchar 和 text：

* 1.varchar 可指定 n，text 不能指定，内部存储 varchar 是存入的实际字符数 +1 个字节（n<=255）或 2 个字节(n>255)，text 是实际字符数 +2 个字节。
* 2.text 类型不能有默认值。
* 3.varchar 可直接创建索引，text 创建索引要指定前多少个字符。varchar 查询速度快于 text, 在都创建索引的情况下，text 的索引似乎不起作用。

**5.二进制数据(\_Blob)**

* 1.\_BLOB和\_text存储方式不同，\_TEXT以文本方式存储，英文存储区分大小写，而\_Blob是以二进制方式存储，不分大小写。
* 2.\_BLOB存储的数据只能整体读出。
* 3.\_TEXT可以指定字符集，\_BLO不用指定字符集。

**6.日期时间类型**

|  |  |
| --- | --- |
| **MySQL数据类型** | **含义** |
| date | 日期 '2008-12-2' |
| time | 时间 '12:25:36' |
| datetime | 日期时间 '2008-12-2 22:06:44' |
| timestamp | 自动存储记录修改时间 |

若定义一个字段为timestamp，这个字段里的时间数据会随其他字段修改的时候自动刷新，所以这个数据类型的字段可以存放这条记录最后被修改的时间。

**数据类型的属性**

|  |  |
| --- | --- |
| **MySQL关键字** | **含义** |
| NULL | 数据列可包含NULL值 |
| NOT NULL | 数据列不允许包含NULL值 |
| DEFAULT | 默认值 |
| PRIMARY KEY | 主键 |
| AUTO\_INCREMENT | 自动递增，适用于整数类型 |
| UNSIGNED | 无符号 |
| CHARACTER SET name | 指定一个字符集 |