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. 启动数据库服务：

Mysqld[--console] //参数为打印执行log

Ps:如果没有没有配置到环境变量需要进入安装目录下../../bin/

1. 连接数据库服务：

Mysql -u‘root’ [-h‘localhost:3306’] -p

1. 断开数据库服务:

QUIT / EXIT

1. 关闭数据库服务：

Mysqladmin -uroot -p shutdown;

<http://www.runoob.com/sql/sql-tutorial.html> // 参考

数据支持扩展函数，同样也支持+,-,\*,/,%等运算符操作

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;

* 删除数据库：

DROP DATABASE [IF EXISTS ] db\_name；

* 使用/切换到指定的数据库：

USE db\_name； //Database changed操作table前必须先use db\_name

* 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()；

* 查看当前状态/及系统配置

SHOW STATUS / SHOW VARIABLES

* 导入其他库中数据表：

SOURCE ‘D:/\*.sql’;

* 创建数据表：

CREATE TABLE [IF NOT EXISTS] t\_name (

col\_name col\_type[(length)] col\_modifier1 col\_mod2 col\_mod3 …,

col\_name1 col\_type[(length)] col\_modifier1 col\_mod2 col\_mod3 …,

[PRIMARY KEY(col\_id)] // 单独声明,通常指定id

. . .)；

//每个属性以空格隔开，每个column以`,`隔开

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；

* 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；

* 查看标的索引结构：

SHOW INDEX 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、删除或修改列及列的属性

* 向数据表中增加字段：(可以指定在哪个字段的前/后面)

ALTER TABLE t\_name ADD [COLUMN] col\_name col\_type[(length)]

-> col\_modifier col\_modifier1...[AFTER | FIRST exist\_name];

* 修改指定列的属性：

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

* ALTER TABLE t\_name CHANGE col\_name new\_col\_name type\_value ;

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 | FIRST;

注意：AFTER，FIRST要在所有属性的最后面声明；

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

Example: 创建一个表；

mysql> CREATE TABLE t\_employee(

-> uid INT(6) AUTO\_INCREMENT COMMENT '員工ID',

-> name VARCHAR(10) NOT NULL COMMENT ‘員工个人姓名’,

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

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

-> job VARCHAR(20) COMMENT ‘員工的工作職位’,

-> salary FLOAT(8,2) DEFAULT 0.00 COMMENT ‘員工的月薪’

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

-> address VARCHAR(60) COMMENT ‘員工的家庭住址’,

-> create\_time TIMESTAMP DEFAULT NOW()

-> );

修改已有的属性；

Mysql> ALTER TABLE t\_employee MODIFY

-> gender TINYINT(1)

-> 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) ...

-> comment ‘间的信息介绍’

-> AFTER age; // AFTER|FIRST 要在cluase的最后面；

增加修改字段的名称及属性：

Mysql> ALTER TABLE t\_employee CHANGE

-> per\_info person\_info varchar(30) ...

-> comment ‘简单的信息介绍’;

注意change 在修改column\_name的同时也一定要修改column\_type,强制

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

VALUES (value1, value2,...valueN );

* INSERT INTO t\_name VALUES

(value1, value2,...valueN );

Ps: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 [as] t1，t2\_name [as] 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：
* 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；

* 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：
* UPDATE user SET password = PASSWORD(“value”) WHERE user = “user”;

Exmple:

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

* Mysqladmin -uroot -padmin password 123456;
* 数据安全连接方案：
* 建议不使用外部ip进行连接，即不建议打开远程连接；
* 使用本地localhost连接；
* 设置其他可访问数据账户；

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

索引分单列索引和组合索引。单列索引，即一个索引只包含单个列，一个表可以有多个单列索引，但这不是组合索引。组合索引，即一个索引包含多个列。

**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 //连接条件

WHERE

conditions

GROUP BY

exist\_column

HAVING

group\_conditions

ORDER BY

exist\_column [ASC | DESC] // 默认ASC

LIMIT

[ offset ], length; //分页查询会用到limit,从offset开始返回length数量

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)。

注意：MySQL 的 WHERE 子句的字符串比较是不区分大小写的。 你可以使用 BINARY 关键字来设定 WHERE 子句的字符串比较是区分大小写的。

**BINARY：**

 区分所查条件与结果字符之间大小写；如：name=”wxling”和name=”Wxling”

SELECT \* from t\_name WHERE BINARY col\_name = “Wxling”; //匹配大小写；

**LIKE** **:**

过滤指定字段含有某些字符的查询结果；“相似过滤”

1. **%**：表示任意 0 个或多个字符。可匹配任意类型和长度的字符，有些情况下若是中文，请使用两个百分号（%%）表示。

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

3. **\_**：表示任意单个字符。匹配单个任意字符，它常用来限制表达式的字符长度语句。

4.  **[]**：表示括号内所列字符中的一个（类似正则表达式）。指定一个字符、字符串或 范围，要求所匹配对象为它们中的任一个。

5. **[^]** ：表示不在括号所列之内的单个字符。其取值和 [] 相同，但它要求所匹配对象 为指定字符以外的任一个字符。

SELECT \* from t\_name 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）需要指定执行排序的字段

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

**GROUP BY:**

语句根据一个或多个列对结果集进行分组，通常配合AVG(),SUM(),COUNT()...等函数使用

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

SELECT name, COUNT(\*) FROM employee\_tbl GROUP BY name;

以name为单位返回不重复name及name数量数据；

**WITH ROLLUP** 可以实现在分组统计数据基础上再进行相同的统计（SUM,AVG,COUNT…）。

SELECT name, SUM(singin) AS singin\_count FROM employee\_tbl GROUP BY name WITH ROLLUP;

**UNION：** [ALL | DISTINCH ] 默认为DISTINCH

UNION 操作符用于连接两个以上的 SELECT 语句的结果组合到一个结果集合中。默认多个 SELECT 语句会删除重复的数据。

SELECT country FROM Websites UNION ALL SELECT country FROM apps;

* 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；**

* **INNER JOIN（内连接,或等值连接）**：获取两个表中字段匹配关系的记录。
* **LEFT JOIN（左连接）：**获取左表所有记录，即使右表没有对应匹配的记录。
* **RIGHT JOIN（右连接）：** 与 LEFT JOIN 相反，用于获取右表所有记录，即使左表没有对应匹配的记录。

例子：现有两张表：t\_user,t\_goods:



* 内联基本语法： //每个字段都可以声明别名，对位获取数据后的参数；

-> 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

-> [cluese1] && [cluese2] //cluese1必填 如t1.id = t2.uid...



-> 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 INNER JOIN t2\_name t2,... //inner可省略

-> ON

-> [ cluase ] //必填 如t1.id = t2.uid...内联条件

-> WHERE

-> [ cluase ] //如前10条记录 id<10



* 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 OUTER JOIN t2\_name t2,...

-> ON

-> [cluase] //如 t1.id = t2.uid...

-> WHERE

-> [cluase] //如前10条记录 id<10



**右连接：**用于获取右表所有记录，即使左表没有对应匹配的记录

-> SELECT

-> t1.col\_name1,t1.col\_name2,...t2.col\_name1,t2.col\_name2,...

-> FROM

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

-> ON

-> [cluase] //如 t1.id = t2.uid...

-> WHERE

-> [cluase] //如前10条记录 id<10



**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;

索引

*索引在MySQL中也叫是一种“键”，是存储引擎用于快速找到记录的一种数据结构。索引对于良好的性能*  
*非常关键，尤其是当表中的数据量越来越大时，索引对于性能的影响愈发重要。*

索引的目的在于提高查询效率

**本质都是：通过不断地缩小想要获取数据的范围来筛选出最终想要的结果，同时把随机的事件变成顺序的事件，也就是说，有了这种索引机制，我们可以总是用同一种查找方式来锁定数据。**

mysql中的primary key，unique，联合唯一也都是索引，这些索引除了加速查找以外，还有约束的功能

**普通索引INDEX：加速查找**

**唯一索引：**

**-主键索引PRIMARY KEY：加速查找+约束（不为空、不能重复）**

**-唯一索引UNIQUE:加速查找+约束（不能重复）**

**联合索引：**

**-PRIMARY KEY(id,name):联合主键索引**

**-UNIQUE(id,name):联合唯一索引**

**-INDEX(id,name):联合普通索引**

* **SELECT** - 从数据库中提取数据
* **UPDATE** - 更新数据库中的数据
* **DELETE** - 从数据库中删除数据
* **INSERT INTO** - 向数据库中插入新数据
* **CREATE DATABASE** - 创建新数据库
* **ALTER DATABASE** - 修改数据库
* **CREATE TABLE** - 创建新表
* **ALTER TABLE** - 变更（改变）数据库表
* **DROP TABLE** - 删除表
* **CREATE INDEX** - 创建索引（搜索键）
* **DROP INDEX** - 删除索引

SQL 对大小写不敏感：SELECT 与 select 是相同的。

**数据类型：**

MySQL中定义数据字段的类型对数据库的优化是非常重要的。

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**：1个汉字＝3个字节 **GBK**：1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(field) from t\_name

**整型**

|  |  |
| --- | --- |
| **MySQL数据类型** | **含义（有符号）** |
| TINYINT(m) | 1个字节  范围(-128~127) 无符号范围(0，255) |
| SMALLINT(m) | 2个字节  范围(-32768~32767) 无符号范围 (0，65 535) |
| MEDIUMINT(m) | 3个字节  范围(-8388608~8388607) |
| INT/INTEGER(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(0~255)。

**2、浮点型**

|  |  |
| --- | --- |
| **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 |
| DECIMA(m,d)L | 定点数，如果M>D，为M+2否则为D+2,依赖于M和D的值 |

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 123.45，但总个数还以实际为准，即5位。

**3、字符串**

|  |  |
| --- | --- |
| **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 |
| TINYBLOB | 二进制字符串，最多255个字符 |
| TINYTEXT | 可变长度，最多255个字符 |
| BOLB | 二进制字符串，最多65535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 |

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' |
| YEAR | 年份 ‘2019’ 取值范围1970-2055 |
| 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 |

主键、外键和索引的区别？

|  |  |  |  |
| --- | --- | --- | --- |
|  | 主键 | 外键 | 索引 |
| 定义： | 唯一标识一条记录，不能有重复的，不允许为空 | 表的外键是另一表的主键, 外键可以有重复的, 可以是空值 | 该字段没有重复值，但可以有一个空值 |
| 作用： | 用来保证数据完整性 | 用来和其他表建立联系用的 | 是提高查询排序的速度 |
| 个数： | 主键只能有一个 | 一个表可以有多个外键 | 一个表可以有多个惟一索引 |