# MySql

### 1.数据库的基本概念

**数据库(DataBase DB)**是一个存储在计算机内的、有组织的、有共享的、统一管理的数据集合。在关系数据库中，数据库表是一系列二维数组的集合，表中的一列称为一个字段，一行称为一条记录。

数据库系统成熟的标志就是数据库管理系统的出现。

**数据库管理系统(DataBase Management System，简称DBMS)**是对数据库的一种完整和统一的管理和控制机制。数据库管理系统的核心是一个用来存储大量数据的数据库。 DBMS能定义数据存储结构，提供数据的操作机制，维护数据库的安全性，完整性和可靠性。

数据库应用程序(DataBase Application)：为了提高数据库系统的处理能力所使用的管理数据库的软件补充。数据库应用程序负责与DBMS进行通信。

**数据库系统有三个主要的组成部分**：**数据库**(用来存储数据的地方)、**数据库管理系统**、**数据库应用程序**。

Oracle、MySQL、DB2都是关系型数据库管理系统。

**MySQL的优势**：运行速度快、免费、容易使用、可移植性、丰富的接口、支持查询语句、安全性。

### 2.实体关系模型

关系数据库的逻辑模型叫做实体(Entity)--关系(Relationship)模型。这个也叫ER。

**①实体关系模型的三要素**：

**实体**：客观存在并可以相互区分的事物称为实体。如：学生、课程。

**属性**：实体所具有的某一特性称为属性，一个实体可以用若干属性来刻画。如学生实体具有学号、姓名等属性。

**关系**：两个实体之间的相关性。如学生与课程之间的关系。

**②实体之间的关系**：

**一对一**：一个实体中的一种情况只与另一个实体中的一种情况有关系。如一个学生实体对应一个学生证实体。

**一对多**：一个实体中的一种情况与另一个实体中的多种情况有关系。如一个班级实体对应多个学生实体，一个学生实体只属于一个班级实体。对应两个表，学生表和班级表，其中学生表中处理学生的基本信息还有一个所属的班级信息。

**多对多**：一个实体中的多种情况与另一个实体中的多种情况有关系。如学生实体对应老师实体， 一个学生实体对应多个老师实体，一个老师实体也对应多个学生实体。对应三个表，学生表、教师表、第三个表中存放用来标识学生的id和表示教师的id。

### 3.数据库的规范化

第一范式：数据库表中的字段都是单一属性的，不可再分。如属性为“姓名年龄”则不符合该范式，该属性还可在分。

第二范式：数据库表中的字段不存在部分函数依赖。

第三范式：数据库表中的字段不存在传递函数依赖。

### 4.SQL语言的分类

SQL(Structure Query Language)：**结构化查询语言**

DQL(Data Query Language)：**数据查询语言**，对应的操作为**select**。

根据SQL语言的不同功能可分为如下四种：

**①DML**(Data Manipulation Language)：**数据操作语言**，对应的操作为**insert、delete、update**。select查询操作也可以归为这类。**用来对表中的数据进行增删改查等操作**。

**②DDL**(Data Definition Language)：**数据定义语言**，对应的操作为**create、drop、alter**。**主要用于建数据库、建表、删除表、修改表的字段等操作**。

**③DCL**(Data Control Language)：**数据控制语言**，对应的操作为**grant、revoke、commit、rollback**。**用来控制对数据库的访问、启动关闭等**。

**④数据库事物处理语言**：实现对数据的交易过程的完整控制。

## 一、MySql的目录结构

注意windows操作系统下，mysql是不区分大小写的，而Linux操作系统下，mysql是区分大小写的。

自己安装的mysql存放的目录：C:\Program Files\MySQL\MySQL Server 5.7。MySQL的默认端口号是3306。

1.bin目录：存放可执行文件

2.data目录：存放数据文件

3.docs目录：存放文档

4.include目录：存储包含的头文件

5.lib目录：存储库文件

6.share目录：错误信息和字符集文件

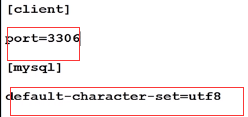
7.**my.ini是配置文件**

直接按Win+R键，然后输入services.msc就可以直接进入电脑的服务窗口，然后找到mysql，双击就可以弹出属性框，就可以看到mysql的存放目录。

## 二、MySql的配置文件

打开my.ini文件，可以查看mysql的配置信息。

client是客户端端口号的地方。



**mysql的编码格式**：注意这里编码格式的写法是utf8而不是utf-8。





注意，若修改了mysql的配置文件，需要在Windows的服务中重启mysql。

## 三、启动与关闭MySql

### 1.在windows操作系统中

右键开始菜单-->计算机管理-->服务和应用程序-->服务，找到MySql进行启动或者关闭。或者在cmd中输入services.msc回车进入。

### 2.在命令提示符窗口

调用net start mysql启动mysql

调用net stop mysql关闭mysql

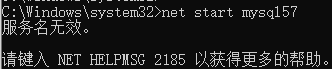
重启mysql：输入mysqld restart

## 四、采用net start mysql启动

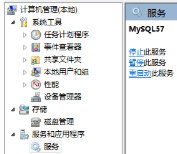
### 1. 提示服务名无效

在cmd中输入net start mysql回车之后提示：**服务名无效，请键入NET HELPING 2185以获得更多的帮助**。当提示服务名称无效时，可能就是mysql的名称写错了，也可能是没有安装mysql。

采用net start+服务名称启动mysql时，这里的服务名称要和windows中的服务名称一致。win+R键然后输入：services.msc就可以直接进入windows的服务窗口，然后查看对应的mysql名称。一般默认的话mysql的名称为MySQL57，自己的是MySQL。如果是MySQL57，则启动的时候是net start mysql57，而不是net start mysql。

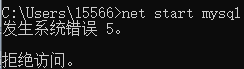


出错原因：net start+服务名称，启动的是在windows中注册过的服务。即在计算机管理-->服务与应用-->服务，该服务窗口中存在的服务。



### 2.提示发生系统错误，拒绝访问

直接在cmd命令提示符窗口输入net start mysql 或net stop mysql会提示：发生系统错误，拒绝访问。



上面提示是因为没有使用管理员权限启动mysql，因此需要使用管理员权限的命令提示符窗口。打开管理员权限的命令提示符窗口：

C盘-->Windows-->System32-->找到cmd.exe，右键以管理员运行。

打开管理员权限的提示符窗口后，输入net start mysql，提示如下：



## 五、重新安装mysql的操作

**①第一步先把原来安装的mysql文件夹中的data的数据备份**，注意一定要备份，不然的话到时候这里面的内容会被清空。这是最重要的。

这个data文件夹一般来说和bin目录等是放在同一地方的。若没有，则在服务中找到mysql，右键属性，查看MySql的属性，找到可执行文件的路径，然后根据这个路径去找data文件夹，找到之后进行备份。**注意同时也把my.ini文件备份**。

**②需要把原来的mysql服务删除**：以管理员身份进入命令提示窗口，然后进入mysql的bin目录中，然后输入：**mysqld –-remove mysql**，然后提示删除成功。





也有的时候提示如下：mysql不存在，这说明计算机中没安装MySQL，则直接进行下一步操作。



**③备份完data中的数据之后，把data文件夹删除**。

**④在cmd中进入mysql的bin目录中输入**：**mysqld --initialize-insecure --user=mysql直接回车，这步是用来初始化data文件夹。**执行完这步这时可以在mysql的安装目录中看到新生成了一个data目录。

注意mysqld--initialize-insecure 这个方法初始化完后，root用户无密码。mysqld--initialize --console这个方法初始化完后，root用户有密码，这个密码是一个随机生成的，在data目录下的.err文件中可以查看到。

**⑤将备份的data文件替换这个新生成的data文件。同时在mysql安装目录中添加.ini文件**。若mysql的目录下原本有.ini(my-default.ini或者my.ini)的文件，则将该文件删除，然后替换成备份的.ini文件。

**注意这个备份的.ini文件要修改一下几个内容**：

**存放路径**：**basedir后面的路径是mysql的存放路径，这个路径用双引号括起来。datadir后面的路径是data文件夹存放的路径**。这两个路径要和存放的路径一致。

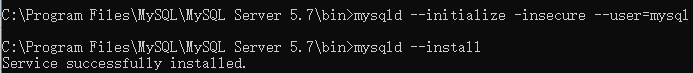




**编码格式：**my-default.ini文件是默认的，使用的是mysql中默认的内容，比如默认数据库的编码格式为latinl。在使用时，需要将编码格式设置为utf8。打开.ini文件，在[client]下添加default-character-set=utf8，在[mysqld]下添加character-set-server=utf8。若原本的.ini文件中有这些则不用添加。

**注意my.ini或者my-default.ini中的内容是用来将mysql注册到windows中的**，所以.ini文件中的内容是很重要的，要保证它的存储位置和编码格式的正确性，不然后面启动MySQL可能失败，或者向数据库中添加中文会出错。

**⑥然后将mysql注册到windows的服务中去**： **mysqld --install**回车，提示Service successfully Installed代表注册服务成功。**注意需要先将my.ini文件中的内容修改完再执行这步，如修改编码格式**。则可以在windows的计算机管理中的服务中可以查看到有MySQL存在。



若注册不成功，则是要在管理员运行的cmd窗口中进行操作。

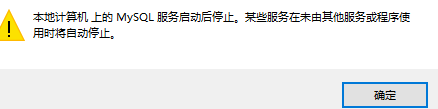


## 六、windows中启动mysql报错

①若要删除mysql服务，在cmd中进入的mysql的bin目录下输入：**sc delete mysql**



②直接在windows的服务中点击启动MySQL，若弹出如下的提示：



则可能是没有把备份的data文件夹替换掉mysql安装目录中刚初始化data的文件夹，或者在安装过程中出现了失误。解决办法：替换data文件夹，或者进行[五、重新安装mysql的操作](#_五、重新安装mysql的操作)。若以管理员的方式打开命令提示符窗口，然后输入net start mysql，启动成功的页面显示如下。



## 七、MySQL的基本操作和规范

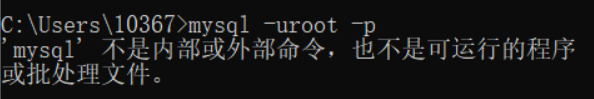
在cmd中对mysql的操作。**注意MySQL中是不区分大小写的**。**数据库中判断相等用一个等号就行**。

### 1.登录MySQL

**mysql –uroot –p直接回车**，然后输入对应的密码。



当运行上面的命令报下面的错误时，要么以管理员的身份在mysql的bin下运行上面的命令，要么将mysql的bin目录配置到系统的环境变量中。



### 2.退出MySQL

登录之后的mysql中，直接输入**exit或者quit或者\q**。



### 3. MySQL常用参数

-h主机名：使用该参数指定主机名或ip，如果不指定默认localhost。

-u用户名：使用该参数指定用户名。

-p密码：使用该参数指定登陆密码。若该字段后面直接跟着一串字符串，则该字符串作为用户密码直接登陆；若后面没有内容则登陆的时候会提示输入密码。

-P端口号：使用该参数指定MySQL服务的端口号，若不指定则默认是3306。

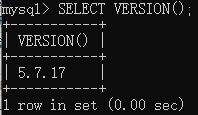


### 4.MySQL常用命令

①显示当前服务器版本：mysql> **SELECT VERSION();**

②显示当前日期：mysql> **SELECT NOW();**

③显示当前用户：mysql> **SELECT USER();**



### 5.MySQL语句规范

**①关键字与函数名称全部大写。(下面为了方便全部写成了小写)**

**②数据库名、表名称、字段名称全部小写。**

**③SQL语句必须以分号结尾。**

**MySQL在windows下是不区分大小写的，在Linux下默认是区分大小写的**。

### 6.MySQL中创建或删除用户

有空再学，内容在书签中。

[..\..\2.和盈课程\1908\5.javaEE2\16MySQL数据库\1.数据库基本概念\MySQL用户管理.ppt](file:///F:\2.和盈课程\1908\5.javaEE2\16MySQL数据库\1.数据库基本概念\MySQL用户管理.ppt)

### 7.MySQL中修改用户密码

有空再学，内容在书签中。

### 8.MySQL中权限管理

有空再学，内容在书签中。

### 9.MySQL的访问控制(重要)

MySQL的访问控制分为两个阶段，**连接核实阶段**和**请求核实阶段**。

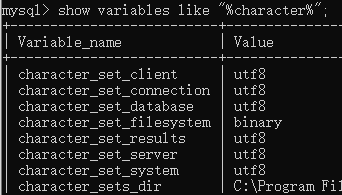
**连接核实阶段**：当客户端向MySQL服务器发送请求连接时，MySQL服务器根据自己的user表校验客户端提供的主机名、用户名和密码是否正确，若校验通过获得连接之后则等待用户发送请求，进入请求核实阶段。若校验不通过则拒绝连接。

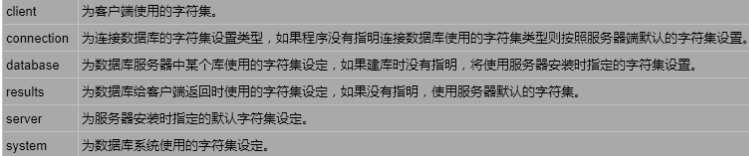
**请求核实阶段**：通过了连接核实阶段之后，用户向MySQL发出操作请求之后，MySQL先检查user权限表中的权限信息，匹配User和Host的值，查看请求的全局权限是否被允许，若权限允许，则对应的操作允许被执行。若请求的权限不被允许，则检查db权限表中的权限信息，匹配User和Host的值，查看请求的数据库级别的权限是否被允许，若权限允许，则对应的操作允许被执行。否则，MySQL继续向下查找，检查tables\_pri权限表中的权限信息，匹配User、Host的值，查看请求的数据表级别的权限是否被允许，若允许则操作可被执行。否则，MySQL继续向下查找，检查columns\_pri权限表中的权限信息，匹配user、host的值，查看请求的列级别的权限是否被允许，若允许，则操作可被执行。若不被允许，则MySQL返回错误信息。

## 八、MySQL中的编码方式

### 1.查看MySQL中的编码方式

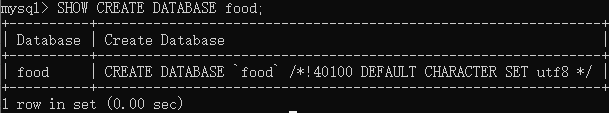
**show variables like “%character%”;**





### 2.查看某个数据库的编码格式

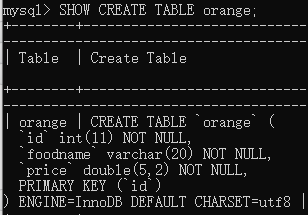
**show create database food(数据库名);**



### 3.查看某个数据库中某个表格的编码格式

**show create table orange(表名);**

**这个需要进入某个数据库之后才能操作，如use food;进入food数据库。**



### 4.创建一个数据库并指定编码格式

**create database person(数据库名) charset=utf8;或者**

**create database person(数据库名) default character set gbk;**



### 5.创建一个表格并指定编码格式

**create table student(表名)(id...name..字段) charset=gbk;**



## 八、DDL的操作

**注意数据库中的desc是关键字，表的表名、列名都不能和关键字重复**。**数据库中的主键不能为空，且不能重复**。

DDL(Data Definition Language)是数据定义语言，**主要用于建数据库、建表、删除表、修改表的字段等操作**。使用的关键字是：**create、drop、alter**。

### 操作数据库：

### 1.查询当前所有的数据库

**show databases；**

### 2.新建一个数据库

创建数据库是在系统磁盘中划分一块区域用于数据的存储和管理。

**create database[ if not exists] food(数据库名);**

[]中的内容代表有或者没有都可以，是可选项。



### 3.删除数据库

删除数据库就是将已存在的库从磁盘删除。且不能恢复

**drop database [if exists] food(数据库名);**



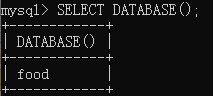
### 4.使用某个数据库

**use food(数据库名);** 这个分号也可以省略。



### 5.查看当前打开的数据库

**select database();**



### 操作数据库中的表：

### 1. 查看数据库中所有的表格

**show tables;** 这个前提条件是使用了某个数据库。

**show tables from pet(数据库);**查看指定数据库中所有的表。



### 2. 在指定数据库中建表

**create table apple(这是表名)**

**( id int primary key,**

**name varchar(20) not null,**

**price double(5,2)**

**);**

设置id为int类型，且为主键primary key。

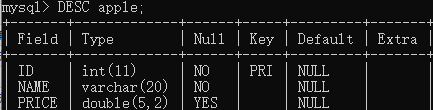
not null：表示不为空。

字符串可以用char和varchar来表示，区别：①char是固定长度，如果长度不够会用空格来补齐；②varchar是可变长，最大长度为规定的长度。

double(5,2)：代表整体长度为5，小数保留两位。

### 3. 查看表结构

**desc apple(这是表名);或者show columns from apple(表名);**



### 4. 修改表结构：

#### ⑴修改表名

**rename table apple(旧表名) to orange(新表名);或者**

**alter table apple rename to orange;**



#### ⑵修改列名的数据类型，不修改列的名字

**alter table orange(表名) modify price(列名) double(5,3);**



#### ⑶同时修改列名和数据类型

**alter table orange(表名) change price(旧列名) color(新列名) varchar(20);**



#### ⑷添加列(字段)

**①添加一列**

**alter table orange(表名) add date(列名) varchar(20);**



这个是默认在末尾添加。

**②添加多列**

添加多列时，用括号括起来，列与列之间用逗号隔开。

**alter table apple add (num int,color varchar(20));**



#### ⑸添加列到第一行

**alter table orange(表名) add foodname(列名) char(5) first;**



#### ⑹添加列到指定的列后面

**alter table orange(表名) add number(新列名) int after color(指定的列名);**



#### ⑺给列名添加约束

这个还不太明白

**alter table apple add primary key (num);**



#### ⑻删除列(字段)

**①删除单列**

**alter table orange(表名) drop date(列名);**



**②删除多列**

删除多列：**drop 列名1,drop 列名2，drop列名3;**

**alter table apple drop num,drop color,drop price;**



**③删除一列之后再添加一列**

删除后添加：**drop 列名1,add 新列名 类型；**

**alter table apple drop price,add num int;**



#### ⑼修改字段的排列位置

**alter table banana(表名) modify price(列名) double(5,2)(列的类型) after name(列名);**注意这个列的类型不能缺少，否则会报错。







### 5. 删除表

**drop table apple(这是表名);**

## 九、DML的操作

DML(Data Manipulation Language)是数据操作语言，**用来对表中的数据进行增删改操作**。**DML使用的关键字是insert、delete、update等**。

### 1.insert增加操作：

**注意在mysql中insert into 的into可以省略，字符串用单引号和双引号都可以。禁止为空的字段一定要赋值，否则报错。**

#### ⑴单条插入：

**①给表格中所有字段插入数据**

**insert into orange(表名) values(1,”橙子”,20.23);**



这里需要注意一下，values()括号中的值的顺序要和表格中对应的列的顺序一致。不一致的话，执行过程会报错。且values()括号中的值的数量要和表格中列名的数量一致。

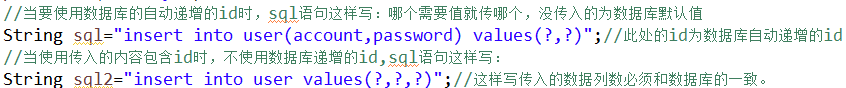


**②给表格中指定字段插入数据**

**insert into orange(price,foodname) values(20,”橘子”);**



注意这里指定的字段顺序可以任意，值的顺序和指定的字段的顺序必须一致。该表格中没有设置值的其它字段，它们的值为系统默认的值。



#### ⑵一次插入多条数据

**insert into orange(foodname,price) values(“org1”,20.2),(“org2”,23);**



这里注意values后面跟着的每条数据用()括号括起来，并且数据之间用逗号隔开。

#### ⑶将一个表复制到另一个表

思路：创建一个结构类似的表，然后查询表格中的数据插入到新表中。

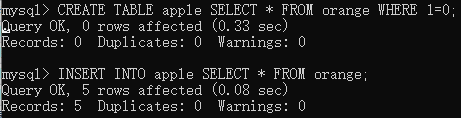
**①方法一：**

**select \* from orange(表名) where 1=0;这个查询的数据结果是空的，但是结果中是有表结构的。**



**步骤一：create table apple(新表名)select \* from orange(表名) where 1=0;这个操作是复制表orange的结构到新表apple中。**

**步骤二：insert into apple select \* from orange ;这个操作是从orange表中查询出数据并插入到新表apple中。**



**②方法二：**

**create table banana(新表名) select \* from orange(表名);**

复制orange表的结构及其数据然后插入到表banana中。



### 2.delete删除操作：

**若不写where条件，则全部删除**。

#### ⑴一个条件的删除

**①delete from banana(表名) where id=5;**

where后面跟着的是条件



**②delete from banana where foodname!=’橘子’;**

**③delete from banana where price is not null;**

#### ⑵多条件的删除

**多个条件之间用and或者or连接。**

**delete from orange where id=4 and price=23;**

**delete from orange where id=2 or price<22;**



### 3.update修改操作：

**若where条件不写，则默认对所有记录进行更新**。

#### ⑴修改单个内容

**update apple(表名) set foodname=”org4” where id=1;**

将apple表中id=1的数据的foodname中的值改为org4。



#### ⑵修改多个内容

**update apple(表名) set id=8,price=24 where id=2;**

多个内容之间用逗号隔开。



### 4.select查询操作：

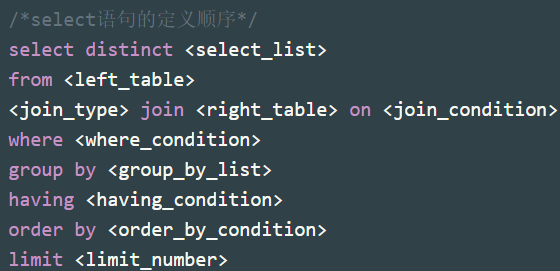
MySQL查询中例子所使用的表：[MySQL中查询使用的表\testdb.sql](file:///F:\1.java笔记（和盈）\3.javaSE基础\MySQL中查询使用的表\testdb.sql)

注意MySQL**查询**默认是不区分大小写的。

MySQL中要查询哪些列是通过select后面的列指定，要查询哪些行是通过where条件指定。

书写查询语句中关键字的顺序：**select top/distinct 字段列表 from 表列表 where 筛选条件 group by 分组字段列表 having 对分组之后得到的结果集筛选 order by 排序字段列表 limit。**

**select语句的定义顺序：**



**select语句的执行顺序：**



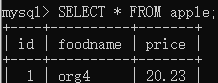
**即执行select语句是先从from开始的。**

#### ⑴无条件/指定条件查询

**①无条件**

**select \* from apple(表名);**

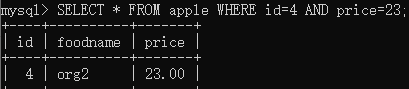
\*代表所有，上面查询的是表中所有的数据。



**②指定条件**

**select \* from apple where id=1;**

**select \* from apple where id=4 and price=23;**



**要特别注意即使where后面的条件不成立，也能得到查询结果，只不过查询结果中的内容都为null，即能得到空表**。

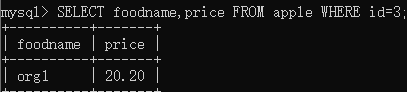
如：select \* from emp where 1=0;



#### ⑵查看指定列的信息

**select foodname(列名),price(列名) from apple(表名) where id=3;**

多个列名之间用逗号隔开。指定列查询的结果只能看到指定列的信息。



#### ⑶多条件查询(and、or、not)

通过逻辑运算连接条件：and 、or 、not。多个条件必须都成立的时候用and。多个条件只需要满足一个的时候用or。将not放在条件的前面，就是对条件的取反。

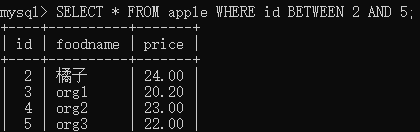
多条件查询和模糊查询的结合：[多条件查询和模糊查询的结合](#_十三、多条件查询SQL语句写法)

**select \* from apple where not name=’org1’;**

#### ⑷between...and...

**select \* from apple(表名) where id between 2 and 5;**

between...and...是**包含边界**的，上面的条件的效果和id>=2 and id<=5的效果是一样的。not between...and ...。



#### ⑸in和not in

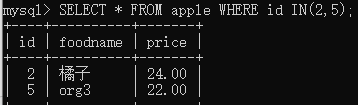
**这个用来筛选“在”或者“不在”某个范围内的结果。**

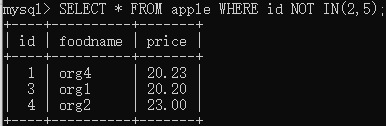
**select \* from apple where id in(2,5);**

**in和not in的后面也可以跟一个查询结果的列表**，如：

**select \* from apple where id in(select \* from apple where price<=30);**

查询的是id为2或5的结果。not in则查出的id既不是2也不是5的结果。





**between...and和in的区别：**

**between n and m表示的是字段的值介于n和m范围之间，包括n和m。而in(a,b,c...)表示的是字段的值只要是a,b,c中的任何一个就可以。**[返回子查询的位置](#_②in关键字的子查询)

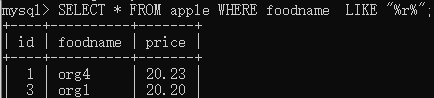
#### ⑹模糊查询：

关键字 **like** 在SQL语句中和通配符一起使用，通配符代表未知字符。SQL中的通配符是 **\_**和 **%** 。其中**\_ 代表只能有一个任意类型的字符，% 代表有不定个任意类型的字符**。

**①like和%一起使用**

**select \* from apple where foodname(列名) like “%r%”;**

**“%r%”表示的是r前面和后面都有未知个字符。**

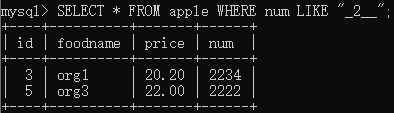


**②like和\_一起使用**

**select \* from apple where num like “\_2\_\_”;**

**“\_2\_\_”表示的是有4个字符，其中第二位为2，2的前面只有一个字符，2的后面只有两个字符。**





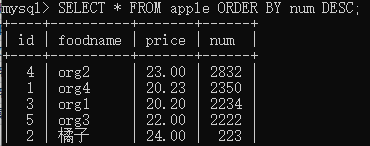
[多条件查询和模糊查询的结合](#_十三、多条件查询SQL语句写法)

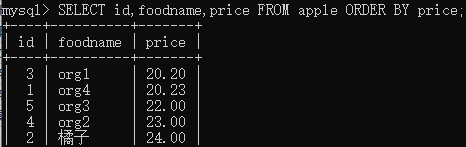
#### ⑺排序order by

在对查询结果排序时需要使用到order by关键字，**默认情况下order by的排序结果是升序**，也可以使用关键字指定排序方式：**asc(升序)、desc(降序)**。**排序时，null看作是最小的。**

**order by 列名1,列名2：先根据列名1进行排序，列名1相同的情况下再进行列名2排序**。

**select \* from apple(表名) order by num(要排序的列名) desc;**





例1：查询员工的信息，按照工资升序排序，如果工资相同，按照名字降序排序。

**select \* from emp e order by e.sal,e.ename desc**;这里sal采用的是升序(没指定asc和desc，默认使用asc)，而ename指定排序方式为降序。

#### ⑻SQL中的组函数

SQL 允许对表中的数据进行计算。SQL 有 5 个内置函数，这些函数都对 SELECT 的结果做操作：

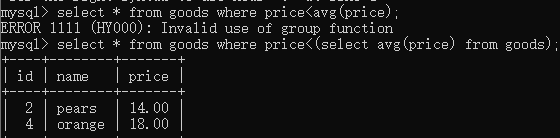


**注意COUNT 函数可用于任何数据类型(因为它只是计数)，而另4个函数都只能对数字类数据类型做计算**。用法如max(列名)、min(列名)等。

**注意：组函数不能直接放在where字句中作为查询条件**。

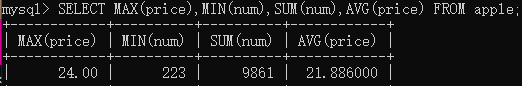
select \* from goods where price<avg(price);错误写法。

**select \* from goods where price<(select avg(price) from goods);**正确写法。



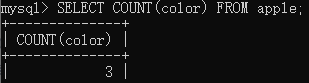
**select max(price),min(num),sum(num),avg(price) from apple;**

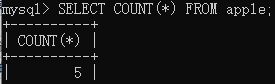
从apple表中查询并计算出price的最大值，num的最小值，num的总和，以及price的平均值。



**当count(指定列名)时，获得的是指定列的总行数，但不包括为null的行数。**

**count(\*)获得的是当前表中所有的行数，不管某列有数值或者空值都包括。**



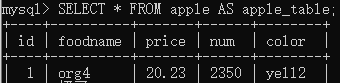


#### ⑼as关键字

在 SQL 语句中，可以为表名称及字段（列）名称指定别名（Alias），别名是 SQL 标准语法，几乎所有的数据库系统都支持。通过关键字 AS 来指定。

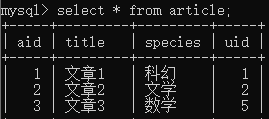
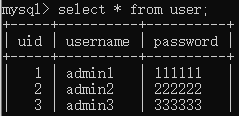
##### ①给表取别名

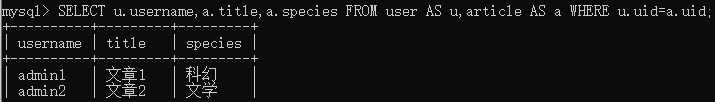
**select \* from apple(表名) as apple\_table(别名);**



**一般对一个表要当作多个表来操作，或者是对多个表进行操作时，才设置表的别名**。

**select u.username, a.title, a.species from user as u, article as a where u.uid=a.uid;**



例1：查询工资在2000-3000之间的员工的名字和所在部门名字。emp表(ename,sal,deptno)和dept表(deptno,dname)。

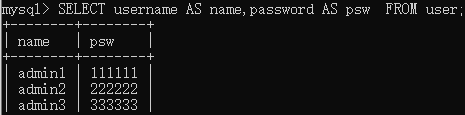
select t1.ename,d.name from (select e.ename,e.deptno from emp e where e.sal between 2000 and 3000) t1,dept d where t1.deptno=d.deptno;

##### ②给字段(列)取别名

给字段取别名可以自定义查询数据返回的字段名称。

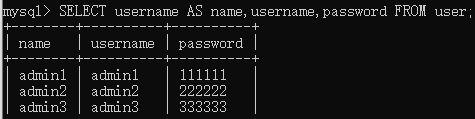
**select username as name,password as psw from user;**

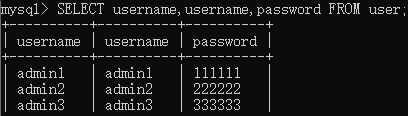
下面查询的结果列名为name和psw而不是username和password。



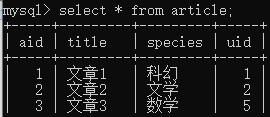
**字段取别名的意义是解决字段名重复的问题，如一个表中的字段被查询两次或多次时**或者**对多个表进行查询时，有相同的返回字段**。

**select username as name,username,password from user;**

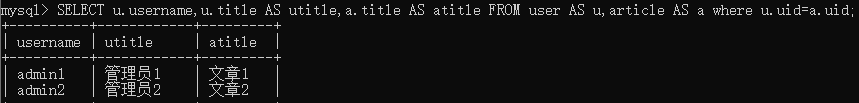


(列名重复)

**select u.username, u.title as utitle, a.title as atitle from user as u, article as a where u.uid=a.uid;**







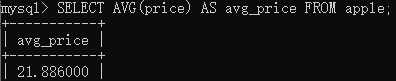
例1：查询员工的编号，名字，岗位，年薪(sal\*12+奖金)。emp表(empno,ename,job,sal,comm)。其中comm是奖金，有的员工没有奖金，即comm的值为null。

select empno,ename,job,sal\*12+**ifnull(comm,0)** year\_sal(列别名) from emp;

这里特别注意comm有的值为null，直接执行sal\*12+comm会导致有的结果为null，因为null参与算术运算的结果都为null，所有要使用ifnull()函数。

##### ③给查询结果取别名

**select avg(price) as avg\_price from apple;**



**通常，定义字段别名的 AS 关键字可以省略**。下面两句的效果是一样的。

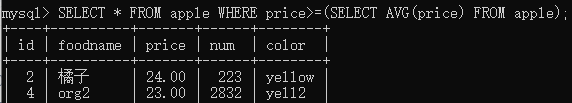
select username as name from user;

select username name from user;

#### ⑽子查询

**一个查询结果作为另一个查询语句的条件。**

**select \* from apple where price>=(select avg(price) from apple);**



##### ①any,some,all关键字的子查询

any和some表示满足其中的任意一个条件就可以，all表示必须满足子查询的所有条件。它们允许对一个查询结果的列表进行比较。

在不使用any、some、all关键字的情况下，不能直接对查询结果的列表进行比较，即下面将all关键字去掉是错误的写法。

例1：查询比部门10的工资都要高的员工的信息。

select \* from emp where sal>**all** (select sal from emp where deptno=10);

这里的子查询select sal from emp where deptno=10的结果是一个列表。where sal>all(...)表示sal要大于子查询结果中的所有数据。

##### ②in关键字的子查询

[超链接到in和not in](#_⑸in和not_in)

##### ③exists、not exists关键字的子查询

exists关键字后面是一个任意的子查询，如果子查询至少返回一行那么exists的结果为true，那么外层查询就会进行查询，否则如果exists没有返回结果，外层查询不会执行。exists和not exists的结果只取决于是否有返回行，与返回内容无关。(不太明白)。

select dname from dept where deptno>200的查询结果如下：



若执行下面的语句：

select \* from emp where exists(select dname from dept where deptno>200);

查询结果如下：从查询结果中可以知道也进行了查询，只不过查询结果为空。



#### ⑾多表关联查询

##### 连接查询：

在连接查询中一定要指定联合条件，否则会出现笛卡尔乘积现象。即查询多张表时一定要有查询的条件。

员工表：emp(empno,ename,job, sal, deptno,mgr,hiredate)

部门表：dept(deptno,dname,loc)

工资级别表：salgrade(grade,losal,hisal)

**①内连接(inner join...on...)**：只有满足（联合）条件的记录才会出现在结果中。

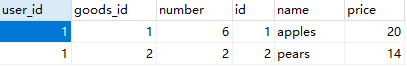
例1：查询员工的名字，岗位，工资，所属部门编号，所属部门的名字。emp表和dept表

**法一：**select e.ename,e.job,e.sal,e.deptno,d.dname from emp as e,dept as d where e.deptno=d.deptno;

**法二：**select e.ename,e.job,e.sal,e.deptno,d.dname from emp e **inner join** dept d **on** e.deptno=d.deptno;

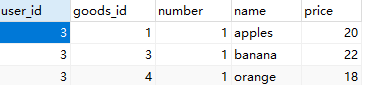
例2：查询cart表中的goods\_id和goods表中的id相同的并且cart表中的user\_id=1的所有信息：cart( user\_id/goods\_id/number)，goods( id/name/price)。

select \* from cart c,goods g where c.goods\_id=g.id and c.user\_id=1;**(给表起别名是直接在表写别名)**



例3：查询cart表中的goods\_id和goods表中的id相同的并且cart表中的user\_id=3的指定信息： cart表中的所有信息和goods表中的name、price信息：

select c.\*,g.name,g.price from cart c,goods g where c.goods\_id=g.id and c.user\_id=3;



**②外连接**：若需要将某张表中所有的记录(符合联合条件和不符合联合条件的所有记录)都要查询出来，即返回结果集中不仅有符合连接条件的记录也包含连接表中不符合连接条件的记录。**外连接可分成左连接和右连接**。

**左连接(left join...on...)**：左边表所有的数据，右边表符合联合条件的记录。

**右连接(right join...on...)**：右边表所有的数据，左边表符合联合条件的记录。

例1：查询员工的名字，岗位，工资，所属部门编号，所属部门的名字 ,没有分配部门的员工只显示员工信息(这句话就说明下面要用e.deptno，而不是d.deptno)。emp表和dept表。

select ename,job,sal,**e.deptno**,d.dname from emp e **left join** dept d **on** e.deptno=d.deptno;

例2：查询员工的名字，岗位，工资，所属部门编号，所属部门的名字 ,没有员工的部门只显示部门编号和部门名字(这句话就说明下面要用d.deptno，而不是e.deptno)。emp表和dept表。

select ename,job,sal,**d.deptno**,dname from emp e **right join** dept d **on** e.deptno=d.deptno;

**③不等连接**：联合条件是不等关系。

例1：查询员工的编号、名字、岗位、工资和工资级别。emp表和salgrade表。

select empno,ename,job,e.sal,s.grade from emp e,salgrade s where e.sal<=s.hisal &&e.sal>=s.losal;

**④自连接**：在一个连接查询中涉及的两张表是同一张表，这种查询称作自连接查询。

例1：team(id,name)表，列出比赛场次，其中比赛不能重复。即如中国队与美国队比赛了不能再继续进行美国队和中国队比赛。

select t1.id,t1.name,t2.id as tid,t2.name as tname from team t1,team t2 where t2.id>t1.id;

##### 其它方式的查询

每个查询的结果表还是一张表，可以用这个结果表参与其它的查询。

例1：查询出在“DALLAS”工作的员工的名字，岗位，工资，工资级别。emp表、dept表、salgrade表。

**法一**：select ename,job,sal,s.salgrade from emp e,dept d, salgrade s where e.deptno=d.deptno and e.sal<=s.hisal and e.sal>=s.losal and d.loc=’DALLAS’;

法二：拆分成两个，查询在“DALLAS”工作的员工的名字，岗位，工资作为表1；再查询员工的名字，岗位，工资，工资级别作为表2。根据表1和表2的联系再查询。

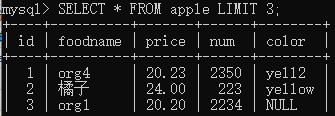
**法二**：select t1.ename,t1.job,t1.sal,t2.grade from(select ename,job,sal from emp e,dept d where e.deptno=d.deptno and d.loc=’DALLAS’)t1,(select ename,job,sal,s.grade from emp e,salgrade s where e.sal<=s.hisal and e.sal>=s.losal) t2 where t1.ename=t2.ename;

#### ⑿limit关键字

**limit关键字一般用来限制起始行的位置。**

**limit [m]n：第一个参数m表示第几条数据**，是可选参数。如果没有指定，默认值是0，表示从第一条记录开始计算。**第二个参数n表示返回的记录的行数**。

**①查询前n条的数据：select \* from apple limit n;或者select \* from apple limit 0,n;**

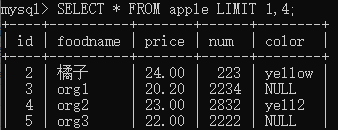


**②从第m+1行开始查询，每次查询n行：**

**法一：select \* from apple limit m,n**

**如limit 1,4表示从第2行开始，每次查询4条数据**

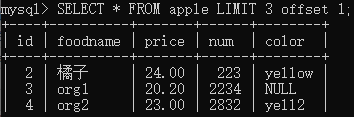
**select \* from apple limit 1,4;**



**法二：select \* from appe limit n offset m;**

**limit n offset m表示从第m+1行开始到第n+m行的数据，n为每次要查询的数据。**

**select \* from apple limit 3 offset 1;表示从第2行开始查询，每次查询3条数据。**



**③查询第n+1到第m行的数据：select \* from apple limit n,m-n;**

在实际应用中一般使用limit来完成分页数据的查询。

例1：假设每页5条数据。

查询第一页：select \* from apple limit 5;

查询第二页：select \* from apple limit 5,5;

查询第三页：select \* from apple limit 10,5;

查询第n页：select \* from apple limit 5\*(n-1),5;

**④查询从第n行开始直到最后一行的数据**：**select \* from apple limit n-1,-1**;第二个参数为-1表示到查询结果的最后一行。

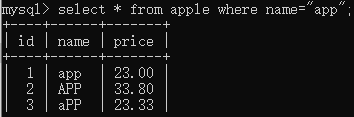
**select \* from apple limit 5,-1;表示查询第6行数据直到最后一行。**

#### ⒀查询时区分大小写

MySQL在执行查询语句时默认是不区分大小写的。执行下面两个语句的效果是一样的。

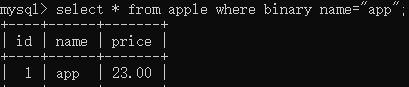
**select \* from apple where name=“app”;**

**select \* from apple where name=“APP”;**



**①区分大小写法一**：**让MySQL查询语句区分大小写**

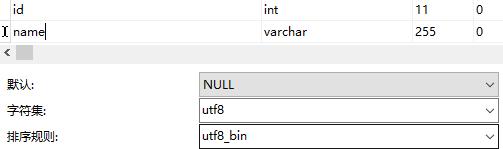
**select \* from apple where binary name=”app”;**



**②在建表时将要区分大小写的加以标识**：

**create table apple(id int primary key,name char(20) binary);**

**③在设计表中设置指定字符集的排序规则为utf8\_bin**：



#### ⒁空值查询

在select语句中使用is null字句查询某字段内容为空的记录。不能使用=号查询空值。若希望查询某字段不为空的记录，则使用 is not null字句查询。

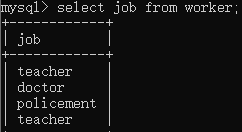
**select \* from apple where name is null;**

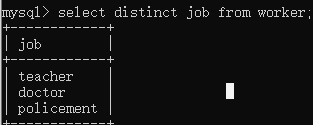
#### ⒂distinct关键字

**在查询时使用distinct关键字修饰列名可以消除重复的数据**。

例1：worker表(id,name,job)中每个员工都有一个job，job可能相同。查询员工表中有几种工作类型。

**select distinct job from worker;**





#### ⒃分组查询(group by...)

##### 用法：

分组查询语法：**group by 列名,...[having 条件表达式]**。其中having+ 条件表达式表示满足条件的结果才会返回。having的用法其实就和where差不多，后面都跟着条件。不使用group by进行分组默认所有的数据都是存放在同一组中。

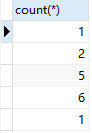
group by关键字通常和组函数count()、sum()、min()、max()、avg()一起使用。

例1：emp表中deptno的值有5种，deptno=10的有2条数据，deptno=20的有5条数据，deptno=30的有6条数据，一种deptno为null值，一种deptno为40的数据。 根据emp表的描述，进行如下操作：

①从emp表中查出总行数，然后再根据deptno分组。

select count(\*) from emp group by deptno;

结果如下所示：从结果中可以直接看出empt表中的deptno的值分成了5组。



②从emp表中查出deptno不为null的总行数。

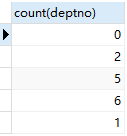
select count(deptno) from emp;结果为14。



③从emp表中查出deptno不为null的总行数，然后再根据deptno分组。

select count(deptno) from emp group by deptno;

结果如下所示：从结果中可以直接看出empt表中的deptno的值分成了5组。从结果中也可以看出不是先从表中查询出deptno不为null的总行数，然后再根据这个结果进行分组。

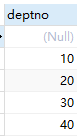


从②和③的结果中说明了一部分问题，就是group by的执行顺序是在count()函数的执行之前。

④下面这两条sql语句的执行结果是一样的，疑问效果是一样的么？

select distinct deptno from emp;

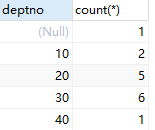
select deptno from emp group by deptno;



⑤查询部门编号以及部门的人数。

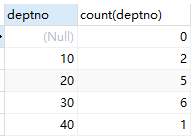
select deptno, count(\*)from emp group by deptno;

结果如下：



select deptno,count(deptno) from emp group by deptno;

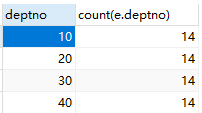
结果如下：



上面两个查询结果中知道：deptno=null时，count(\*)和count(depntno)分别根据deptno分组得到的结果分别为1和0。这是因为根据count(\*)得到总人数的结果是14，而根据count(deptno)得到deptno不为null的总人数结果是13。然后对deptno进行分组当deptno=10、20、30、40对应的人数确定后，deptno=null的人数在count(\*)的记录中就还剩1个人，而在count(deptno)中就没有剩余，则为0。

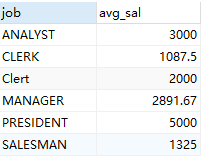
下面这种查询出来的结果是错误的，不太明白为什么，难道分组查询只能在同一表中进行，不能将不同表中的数据查询出来再分组？。

select d.deptno,count(e.deptno) from emp e,dept d group by deptno;



**例2**：查询岗位及岗位工资。

select job,round(avg(sal),2) avg\_sal from emp group by job;



##### 注意事项：

**①**组函数不能直接作为where条件使用，但是可以直接作为having的条件使用。

如：select \* from emp where sal<avg(sal)这种写法是错误的

正确写法：select \* from emp where sal<(select avg(sal) from emp)

**②**select语句使用group by分组后，**在select后面的查询列就只能包含分组的列和组函数数据**，**不能查询任意列的数据**。

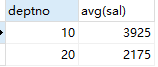
如：查询部门编号和人数select ename,deptno,count(\*) from emp group by deptno。这里的ename没有意义，在MySQL中这样查询虽然不会报错，但是查询的结果是错误的。在Oracle中这样查询会直接报错。

**③**where关键字和having关键字都可以作为查询的过滤条件，where放在分组之前用来过滤记录，having放在分组之后用来过滤分组。Where排除的记录不再包括在分组中。

**组函数的条件放在having子句中**。(group by...having...)

**例1**：查询平均工资高于2000的部门的编号和平均工资。emp表(deptno,sal)。分析：相当于查询出每个部门的编号以及每个部门的平均工资(部门的编号和平均工资根据部门编号进行分组)，在查询出部门平均工资大于2000的部门。

**select deptno,avg(sal) from emp group by deptno having avg(sal)>2000**;



**select...from...where...普通条件...group by... having...组函数条件**。

**例2**：查询平均工资高于2000的岗位、平均工资，有奖金的员工不计算在内。

**select job,avg(sal) from emp where comm is null or comm =0 group by job having avg(sal)>2000**；

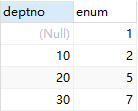
使用group by字句可以对多字段分组。没明白是怎么分组的。

select deptno,job,max(sal) from emp group by deptno,job

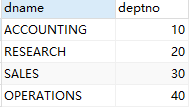
**例3**：查询部门名字，员工人数，没有员工的人数显示0。emp(deptno)，dept(deptno,dname)。

首先查询出部门员工人数：需要查询出每个部门编号以及总人数，然后根据部门编号进行分组。

select deptno,count(\*) enum from emp group by deptno。



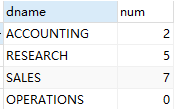
然后从dept表中查询出如下的数据：select dname,deptno from dept



最后从上面两个查询结果中查询数据。采用外连接的方式：

select dname,**ifnull(t1.enum,0)** num from dept d left join (select deptno,count(\*) enum from emp group by deptno) t1 on d.deptno=t1.deptno;

要注意根据条件d.deptno=t1.deptno一一对应时，d表中deptno=40时，t1表中deptno没有40，这时查询的限定条件d.deptno=t1.deptno不成立，但是有查询结果，只不过这时的结果为null，因此t1.enum的值就为null，所以采用ifnull(t1.enum,0)将结果变为0。在查询中，即使查询条件不成立，仍然有查询结果，只不过查询结果为null值。



#### ⒄其它查询

select \* from emp where (sal,comm)<=>(select sal,comm from emp where ename='BLANK');

查询的条件表示绿色部分的sal、comm和蓝色部分查询的列sal、comm的值一一对应，必须同时满足。

上面的sql语句和下面这个sql语句的效果是一样的。

select \* from (select sal,comm from emp where ename='BLANK') t1,emp e where e.sal<=>t1.sal and e.comm<=>t1.comm;

#### ⒅union合并查询结果

**union 和union all可以用来合并两个或多个select语句的结果集， 这些select语句查询的结果集必须列的数据类型一致、顺序和数量一致。其中，union合并之后可以去除重复的数据，同时也会根据字段的顺序进行排序。union all则不会去除重复数据，而是直接将查询结果连接起来。 在执行效率上，union all要比union更快**。

select ename,job from emp where deptno=10 **union** select ename,job from emp where sal between 1000 and 3000;

#### ⒆使用正则表达式查询

MySQL中使用**regexp**关键字指定正则表达式的字符匹配模式。

**例1**：查询员工名字以SM开头的员工。

select \* from emp where ename **regexp** ‘^SM’;

**例2**：查询员工名字以S开头以H结尾的员工。

select \* from emp where ename regexp ‘^S.+H$’;

**例3**：查询员工名字中包含‘ON’的员工。

select \* from emp where ename regexp ‘ON’;

**例4**：查询员工名字中包含‘O’或者’N’的员工

select \* from emp where ename regexp ‘[ON]’;



### 5.增删改查的总结：

**⑴查询**：select 列名1,列名2 from 表名 where....

**⑵增加**：insert into 表名(列名) values(指定值)

**⑶删除**：delete from 表名 where....

**⑷修改**：update 表名 set 列名=值,列名=值,...where....

## 十、MySQL中的数据类型

### 1.整型

MySQL中整型可分为五类：tinyint、smallint、mediumint、int、bigint。

**①tinyint**

占1个字节。存储范围：有符号值为-128到127(-2**7**到2**7**-1)；无符号值为0到255(0到2**8**-1)。

**②smallint**

占2个字节。存储范围：有符号值为-32768到32767(-2**15**到2**15**-1)；无符号值为0到65535(0到2**16**-1)。

**③mediumint**

占3个字节。存储范围：有符号值为-2**23**到2**23**-1)；无符号值为0到2**24**-1。

**④int**

占4个字节。存储范围：有符号值为-2**31**到2**31**-1；无符号值为0到2**32**-1。

 int(11)中这个11表示的是该数据类型指定的显示宽度，数值的位数小于指定的宽度时，会由空格填充，如果插入大于显示宽度，只要该值不超过该类型整数的取值范围，数值依然可以插入而且能够显示出来。int(m)形式，这个m是显示宽度，不能限制数据取值范围和占用空间。

**⑤bigint**

占8个字节。存储范围：有符号值为-2**63**到2**63**-1；无符号值为0到2**64**-1。

### 2.浮点型和定点数类型

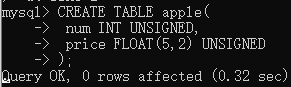
float(M,D)：其中M表示数字总位数，D表示小数点后面的位数。若M和D被省略了，则根据硬件允许的限制来保存值。**单精度浮点数(float)精确到大约小数点后7位**，存储的字节数为4字节。

double(M,D)：如double(5,2)表示总位数为5为，其中小数点后面保留2位。存储的字节数为8字节。

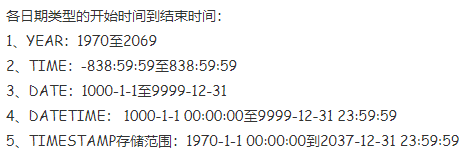
**decimal(M,N)**：定点数以字符串形式存储，在对精度要求比较高的时候(如货币,科学数据等)使用decimal的类型比较好。存储的字节数为M+2个字节。numeric的功能相当于decimal。decimal即可用来定义整数也可定义小数，实际上存储是以字符串的形式。

### 3.unsigned关键字

**UNSIGNED关键字表示无符号值，即正数**。常和整型和浮点型一起使用。



### 4.日期时间型



#### ⑴year

year存储时只需一个字节。若以4位字符串或者4位数字格式表示YEAR，则范围为1901~2155。若以2位字符串表示 "00"~"69" 代表2000~2069 "70"~"99" 代表1970~1999。若以2位数字表示YEAR 1-69代表 2001~2069 70~99 代表 1970~1999。

#### ⑵time

日期格式为hh:mm:ss，即时分秒。存储需求为3个字节。日期范围为-838:59:59~838:59:59。

#### ⑶date

日期格式为yyyy-mm-dd，即年月日。存储需求为3个字节。

#### ⑷datetime

日期格式为yyyy-mm-dd hh:mm:ss，即年月日时分秒。存储需求为8字节。

#### ⑸timestamp

日期格式为yyyy-mm-dd hh:mm:ss，即年月日时分秒。存储需求为4字节。timestamp可以作为时间戳，用来记录数据最近一次更新的时间

**timestamp和datetime的区别**：存储字节大小不同，表示的范围不同。datetime在存储日期数据时，按照实际输入的格式存储，即输入什么存储什么，与时区无关。而timestamp值的存储以UTC格式保存，存储时对当前时区进行转换，检索时再转回当前时区。

当存储范围较大的日期最好用datetime。

### 5.文本字符串类型

**①char(M)**

为固定长度字符，M为存储的字节数，M的范围为0到255。

char(5)存储“abc”时，剩余两位用空格补齐。char(3)存储“abcde”时，会报错。

**②varchar(M)**

为可变长度字符。存储L+1个字节，其中L<=M并且0<=M<=65535。实际占用的空间为字符串的实际长度+1个字节。

varchar(20)存储“abc”时，只存储abc而不添加空格，且存储的字节大小为3字节。当存储数据的长度超过M指定的长度时会报错。

char是固定长度，它的处理速度比varchar快，缺点是浪费存储空间。对于MyISAM引擎，最好使用固定长度的数据列，这样可以使表静态化，加快表格检索速度。对于InnoDB引擎，最好使用可变长度的数据列，因为InnoDB的数据表的存储格式不区分固定长度和可变长度，而 varchar按照实际长度存储，对于磁盘I/O和数据存储总量比较好。

text类型保存的是非二进制类型的字符串。

**③tinytext：**存储L+1个字节，其中L<28。非常小的非二进制字符串。

**④text：**存储L+2个字节，其中L<216。

**⑤mediumtext：**存储L+3个字节，其中L<224。中等大小的非二进制字符串。

**⑥longtext：**存储L+4个字节，其中L<232。大的二进制字符串。

### 6.二进制字符串类型

**①binary(M)**：为固定长度二进制字符串，存放M个字节

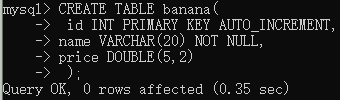
**②varbinary(M)**：可变长度二进制字符串，存放M+1个字节

**③tinyblob、blob、mediumblob(M)、longblob(M)**等。blob是一个二进制对象，存储的是二进制字符串。

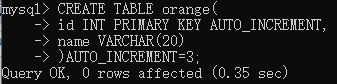
## 十一、自动编号和约束

### 1.自动编号(auto\_increment)

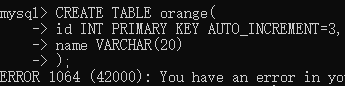
自动增加可以保证记录的唯一性。**自动编号必须和主键primary key组合使用，否则报错**。**自动编号的类型必须为整型，也可以是浮点型，若为浮点型则小数点后必须为0，若float(5,0)。默认情况下，起始值为1，每次增量为1**。



#### ⑴在创建表时指定自动编号的起始值



注意写成下面这样是错误的：



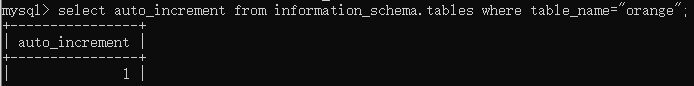
#### ⑵修改指定表自动编号的起始值

**alter table orange(表) auto\_increment=5;**



**注意修改的起始值必须大于auto\_increment的自增值，否则设值的起始值无效**。如当起始值设置成0.5，而自增值还是1时，查看auto\_increment的起始值如下：

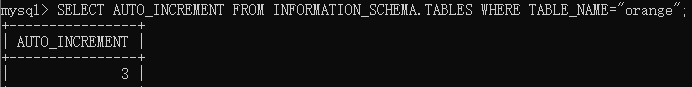




#### ⑶查看指定表自动编号的起始值

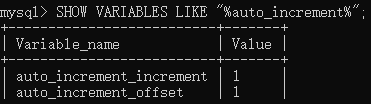
**select auto\_increment from information\_schema.tables where table\_name=“orange”;**





#### ⑷查看自动编号相关的变量设置

**show variables like “%auto\_increment%”;**

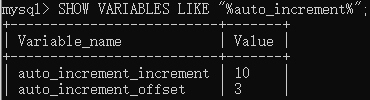


其中auto\_increment\_offset的值为自动编号的起始值，而auto\_increment\_increment的值为自动编号每次自动增加的值。

#### ⑸修改自动编号的变量





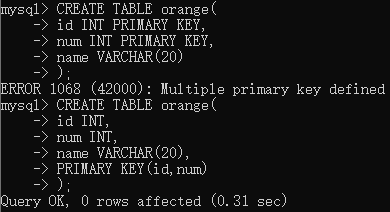


**注意，当设置的auto\_increment\_offset的值大于auto\_increment\_increment的值时，auto\_increment\_offset设置的值无效**。修改自动编号的变量和修改指定表自动编号的起始值两者的联系好像矛盾，不太明白。

### 2.主键约束

**主键：用来唯一的标识表中的每一条记录。自动增加(auto\_increment)定义的字段必须被定义成主键(primary key)，但被定义成主键的字段不一定要被定义成自动增加**。主键自动为not null。若主键没有被定义成自动编号，则必须为其赋值，若定义了自动编号则可以不用自己赋值。**每张数据表只能存在一个主键**。

组合主键：多字段主键采用primary key(列名1，列名2...)的方式创建。

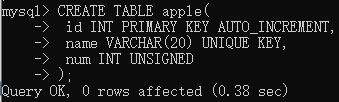


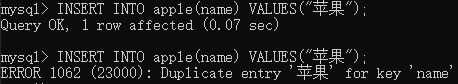
### 3.唯一约束

唯一约束(unique key)可以保证记录的唯一性。**唯一约束的字段可以为空值(null)。每张数据表可以存在多个唯一约束**。

**unique key 和primary key的区别**：**主键是一张数据表只能有一个，不能为空，但是唯一约束可以存在多个，而且可以存在空值**。

注意：**唯一约束，可以有空值，但只能有一个空值**。

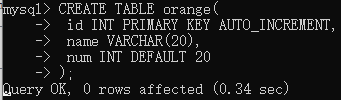


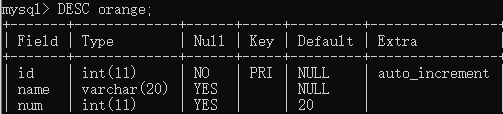


上面将apple表的name属性设置成了unique key，所以name的值不能重复创建相同的，当向表中插入一条name为“苹果”的记录时，可以成功插入，当插入第二条name为“苹果”的记录时，则报错。

### 4.默认约束(default constraint)

当插入记录时，若没有明确为字段赋值，则自动赋予默认值。





### 5.外键约束

#### ⑴外键的概念

外键(**foreign key**)用来在两个表之间建立连接，可以是一列或多列。一个表可以有一个或多个外键。**外键可以为空值，如果不为空，那么值应该对应另外一张表中的某个主键值**。**外键对应表中的一个字段，可以不是本表的主键，但对应另外一个表的主键**。使用外键可以保持数据的完整性和一致性。

**例**：有部门(dept)表，其中的主键是id。有员工(emp)表，员工表中有字段deptId。设置emp表中的字段deptId为外键和dept表中的id字段关联。此时dept表称为主表(父表)，emp表称为从表(子表)。

注意：定义外键后，不允许删除父表中与子表有关联关系的行。子表中被定义为外键的字段的值要么为空，要么和父表中主键的值相匹配。父表中的参照列必须是主键，否则不能作为参照列。子表中插入数据时，父表中的主键必须已经存在数据。不能直接删除父表。

#### ⑵注意事项：

**①**父表和子表必须使用相同的存储引擎 InnoDB。

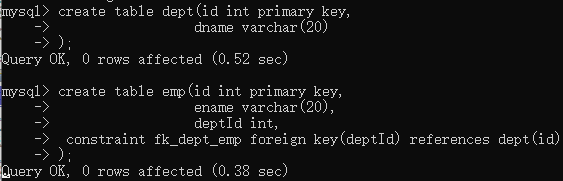
**②**外键列和参照列必须具有相似的数据类型。其中数字的长度和是否有符号位必须相同；而字符的长度可以不同。

**③**外键列和参照列必须创建索引。如果外键列不存在索引的话，MySQL将自动创建索引。

#### ⑶增加外键

**创建外键的语法规则**：**constraint 外键名 foreign key(子表的列) references 父表名(父表的主键列)。外键名一般命名为fk\_xxx。**

**①创建表格时增加外键**



**②使用alter语句增加外键**

语法规则：alter table 子表名 add constraint 外键名 foreign key(子表中的列名) references 父表名(父表的主键列名)[on delete cascade|set nul];

alter table emp add constraint fk\_dept\_emp foreign key(deptId) references dept(id) [on delete set null];

on delete后面跟的是删除策略，是采用cascade或者set null的方式删除。

#### ⑷删除外键

对于定义的外键，若不再需要可以删除。语法规则：

alter table 子表名 drop foreign key 外键名;

alter table emp drop foreign key fk\_dept\_emp;

### 6.非空约束

非空约束(not null constraint)指字段的值不能为空。语法为：字段名 数据类型 not null。

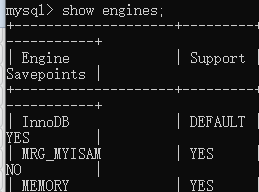
create table apple(id int primary key,name varchar(20) **not null**);

## 十二、数据库存储引擎

 数据库管理系统（DBMS）使用数据引擎进行创建、查询、更新和删除数据的操作。不同的存储引擎提供不同的存储机制。

### 1.查看系统所有存储引擎

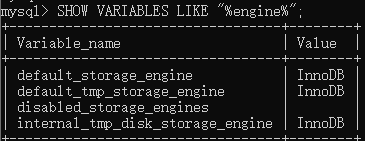
**show engines;**



MySQL的存储引擎有InnoDB、MyISAM、Memory、Merge、Archive、FeDerated、CSV、BLACKHOLE。有空再深入学习一下[..\..\2.和盈课程\1908\5.javaEE2\16MySQL数据库\2.标准sql\MySQL数据库基本操作.ppt](file:///F:\2.和盈课程\1908\5.javaEE2\16MySQL数据库\2.标准sql\MySQL数据库基本操作.ppt)。默认的存储引擎是InnoDB。

### 2.查看默认存储引擎

**show variables like “%engine%”;**



### 3.修改表的存储引擎

**alter table banana(表名) engine=InnoDB(存储引擎);**

### 4.指定存储引擎

⑴创建表格的同时指定存储引擎

create table 表名(

..............

)engin=存储引擎;

## 十三、多条件查询和模糊查询的结合

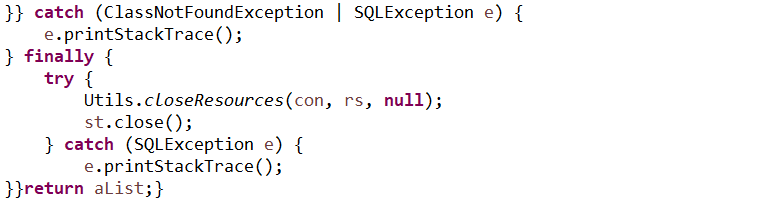
采用字符串拼接方式的SQL语句是非常繁琐的，中间有很多的单引号和双引号的混用，极易出错。要特别注意下面每条拼接的sql语句，双引号和and之间必须空一格，若不空格的话，拼接后的语句会变成下面这样，会报错。



如下例子：

**多条件查询sql语句的写法例子：**





在进行多条件查询时，若查询的条件的数量是不确定的，可以在查询的sql语句后面写一个条件**1=1**，然后多余的条件则采用字符串拼接的方式拼接到该sql语句后面。**"select \* from person where 1=1 "**。注意采用拼接字符串的方式写sql语句只能用Statement而不能使用PreparedStatement。这个sql语句写成StringBuilder类型的，后面要拼接的条件采用StringBuilder的append()方法进行拼接。最后再将该StringBuilder类型的sql转成String类型的sql作为参数传入。

## 十四、MySQL的运算符

每个数据库中都有一个dual表，该表是虚拟表，这个表不是用来存储数据。对于运算的测试可以借助于这个dual表。

### 1.算术运算

+、-、\*、/、%

### 2.比较运算

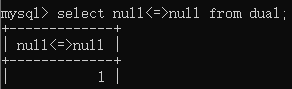
=、<=>安全的等于、<>不等于(也可用！=表示)、<=、>=.......

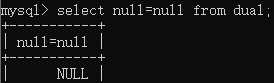
**比较运算的结果是boolean值，用0和1表示，0表示false，1表示true**。

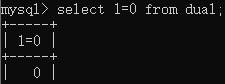
判断数据库表中的结果是否是null值，不能通过null=null来判断，null=null的结果还是null，null！=null的结果也是null。**判断是否是null值要用is null或者<=>判断，判断一个值是否不为null则用is not null判断**。

**<=>代表安全的等于**，它的操作和等于号一样，都可以用来判断是否相等，但是=不能用来判断null的值，而<=>则可以用来判断null的值。null<=>null的结果是1，1表示的是true，0表示的是false。

**null参与任何运算的结果都是null**。







### 3.逻辑运算

逻辑运算的结果为1(true)和0(false)。与(and、&&)，或(or、||)，非(not、！)，异或(xor)。

### 4.位运算

|、&、^、>>、<<等。

## 十五、MySQL中的函数

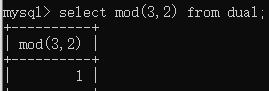
MySQL中系统提供的函数可以直接参与sql语句操作。

### 1.数学函数

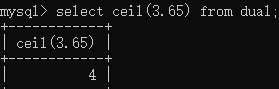
⑴abs(x)：返回x的绝对值。pi()：返回圆周率3.141593。

⑵sqrt(x)：返回非负数x的平方根。pow(x,y)：返回x的y次方。

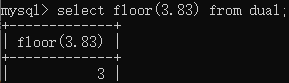
⑶**mod(x,y)**：返回x被y除后的余数。exp(x)：返回e的x次方。



⑷**ceil(x)**：返回大于等于x的最小整数值。返回的是bigint类型。



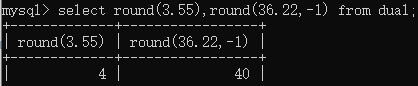
⑸**floor(x)**：返回小于等于x的最大整数值。返回的是bigint类型。

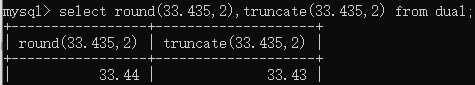


⑹**round(x)**：返回最接近于参数x的整数，对x进行四舍五入。

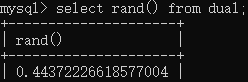
**round(x,y)**：返回最接近于x的数，保留到小数点后y位，若y为负数，保留x到小数点左边y位，会进行四舍五入。

**truncate(x,y)**：保留到小数点后y位，不进行四舍五入。

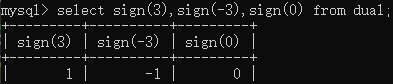




⑺**rand()**：返回0到1之间的随机数。



⑻**sign(x)**：返回x值的符号。正数返回的是1，负数返回的是-1，0返回的是0。

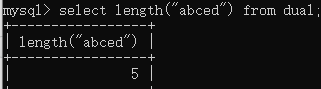


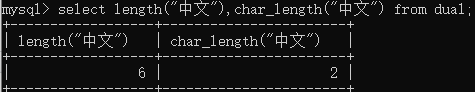
### 2.字符串函数

**⑴length(str)**：返回字符串str的字节长度。

**char\_length(str)**：返回字符串str的字符长度。

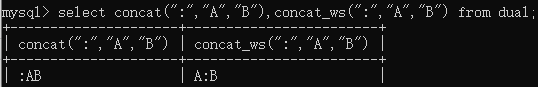
(数据库的编码格式已经修改为utf-8)，在utf-8编码中一个中文字符占3个字节，一个英文占1个字节。gbk编码中一个中文字符占2个字节，一个英文占1个字节。



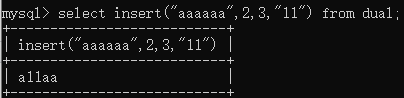


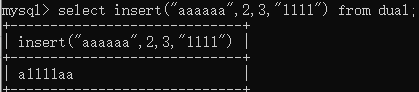
**⑵concat(s1,s2,...)**：将字符串s1、s2...连接起来然后返回。

**concat\_ws(x,s1,s2,...)**：将x作为分隔符，然后将s1、s2...等连接起来返回。

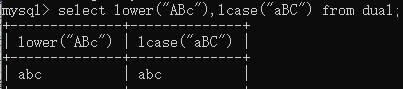


**⑶insert(s1,x,len,s2)**：将字符串s1从第x个位置开始被s2替换掉len个字符，然后返回新s1字符串。注意当字符串s2的长度小于或大于len时的情况。

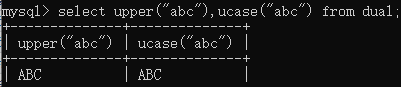




**⑷lower(str)**和**lcase(str)**都是将字符串str中的字符全部变成小写。

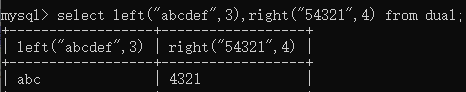


**⑸upper(str)**和**ucase(str)**：将字符串str中的字符转成大写字母。



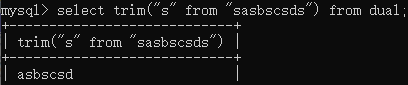
**⑹left(str,n)**：返回字符串str从左边开始数的n个字符。

**right(str,n)**：返回字符串str从右边开始数的n个字符。

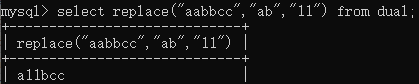


**⑺ltrim(str)、rtrim(str)、trim(str)**：分别删除字符串str左边、右边、两边的空格。

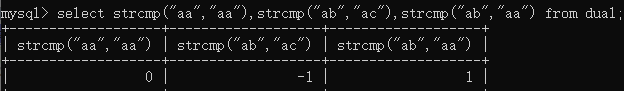
**trim(s from str)**：删除字符串str两边符合s的字符串。



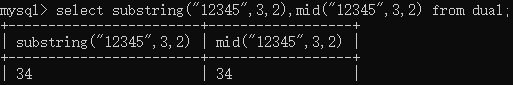
**⑻replace(str,s1,s2)**：将字符串str中的s1字符串替换成s2字符串。



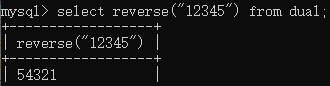
**⑼strcmp(s1,s2)**：比较字符串s1和s2，如果完全相同则返回0，如果s1小于s2则返回-1,否则返回1。



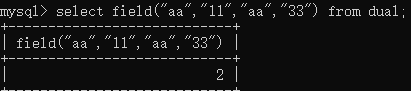
**⑽substring(str,n,len)**和**mid(str,n,len)**：返回字符串str中起始位置为n，长度为len的字符串。



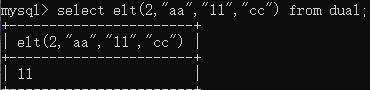
**⑾reverse(str)**：将字符串反转。



**⑿field(str,s1,s2,s3...)**：返回字符串str在指定s1、s2、s3...列表中的位置。



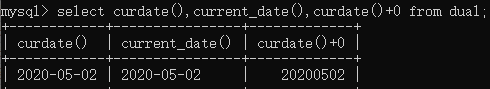
**⒀elt(n,s1,s2,s3...)**：返回指定位置n上的字符串。



### 3.时间和日期

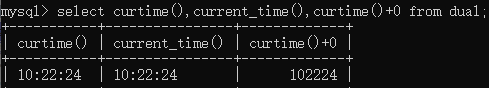
**⑴curdate()、current\_date()**：返回当前的日期，即年月日。默认格式为yyyy-mm-dd。

**curdate()+0**：以yyyymmdd的形式返回当前日期。这个是数字0。



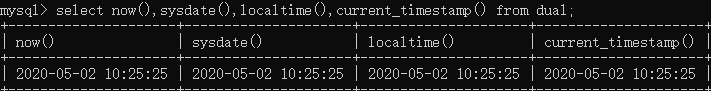
**⑵curtime()、current\_time()**：返回当前的时间，即时分秒。默认格式为hh:mm:ss。

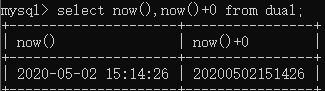
**curtime()+0**：以hhmmss的形式返回当前的时间。这个是数字0。



**⑶now()、sysdate()、localtime()、current\_timestamp()**：返回当前的日期和时间。格式为YYYY-MM-DD hh:mm:ss。

now()+0：以yyyymmddhhmmss的形式返回当前的日期时间。

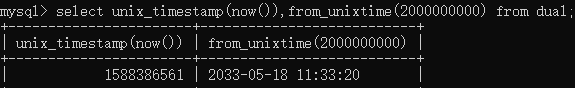




**⑷unix\_timestamp(date)**：返回一个unix时间戳。从‘1970-01-01 00:00:00’GMT（格林威治时间）到date之间的秒数。

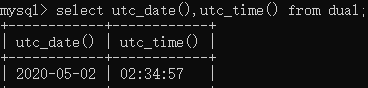
**from\_unixtime(num)**：将unix时间戳转换成普通格式的时间。即将num毫秒数转换成对应的时间。

上面两个函数互为反函数。



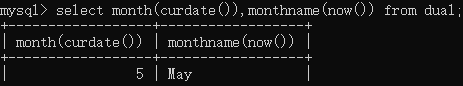
**⑸utc\_date()**：返回当前UTC(世界时间标准)的日期。

**utc\_time()**：返回当前UTC(世界时间标准)的时间。



**⑹month(date)**：返回日期对应的月份。

**monthname(date)**：返回日期对应的月份的英文名称。

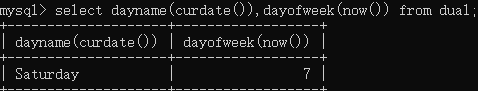


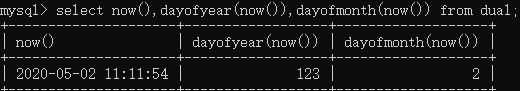
**⑺dayname(date)** ：返回日期对应的工作日的英文名称。

**dayofweek(date)**：返回日期对应的天在一周中的位置。星期日的位置为1，星期一的位置为2，依次类推，星期六的位置为7。范围是1-7。

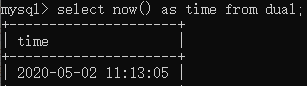
**dayofyear(date)**：返回日期对应的天处于一年中的第几天。范围是1-366。

**dayofmonth(date)、day(date)**：返回日期对应的天处于一个月中的第几天。范围是1-31。





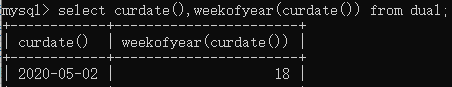
注意可以给函数的结果取别名，但不能将函数的结果取的别名直接作为另一个函数的参数。





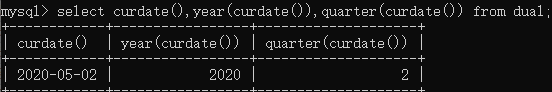
**⑻weekday(date)：**返回date日期中天的索引。0代表周一，1代表周二，6代表周日。

**weekofyear(date)**：返回日期对应的天天处于一年中的第几周。



**⑼year(date)**：返回日期中的年份。

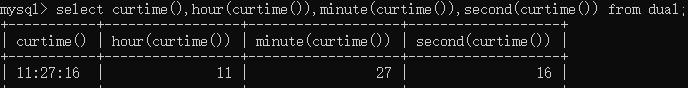
**quarter(date)**：返回日期所对应的月份处于第几季度。每三个月为一个季度。



**⑽hour(time)**：返回时间对应的时。

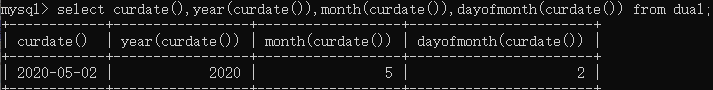
**minute(time)**：返回时间对应的分。

**second(time)**：返回时间对应的秒。

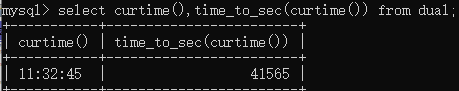


**⑾单独获得日期的年、月、日**。

**year(date)、month(date)、dayofmonth(date)或者day(date)**。

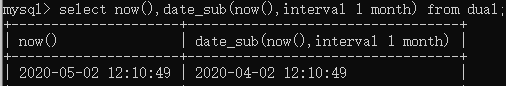


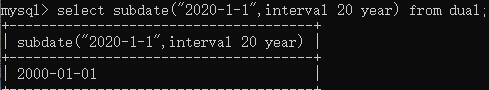
**⑿time\_to\_sec(time)**：将时间转换成秒。公式为hour\*60\*60+minute\*60+second。



**⒀date\_sub(date,INTERVAL expr type)、**

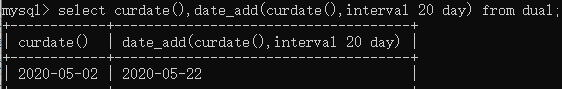
**subdate(date,INTERVAL expr type)**：得到date日期减去指定类型的时间之后的日期。这里的type代表的类型可以是day、year、month、hour、minute、second、quarter、week、year\_month、day\_hour等。interval代表间隔，interval 3 month代表间隔3个月。

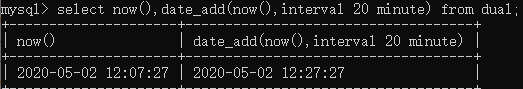




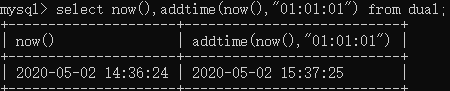
**⒁date\_add(date,INTERVAL expr type)、**

**adddate(date,INTERVAL expr type)**：得到date日期加上指定类型的时间之后的日期。

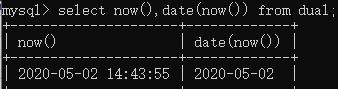




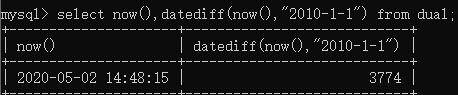
**⒂addtime(expr1,expr2)**：将expr2中的时间添加到expr1中。expr1是一个时间或时间日期表达式。expr2是一个时间表达式，不能用interval 表示。没有time\_add()函数。



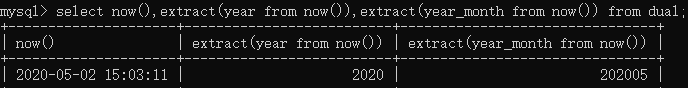
**⒃date(expr)**：得到日期或者日期表达式expr中的日期部分。



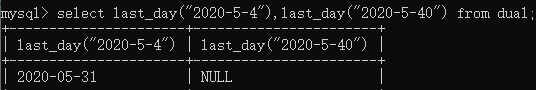
**⒄datediff(expr,expr2)**：得到起始时间expr和结束时间expr2之间的天数，前者减去后者的天数。只计算年月日个这日期部分，不考虑时分秒。



**⒅extract(type from date)**：从date日期或日期时间中截取指定类型的时间。



**⒆last\_day(date)**：返回date日期或者日期时间中对应的月份的最后一天的日期值。若参数无效则返回null。



### 4.加密和解密函数

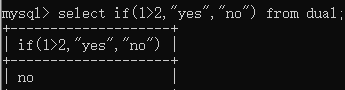
不太重要。

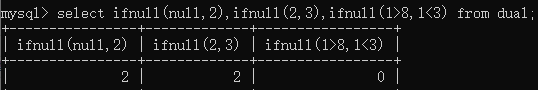
### 5.其它函数

**⑴条件判断函数**：

**if(expr,v1,v2)**：如果表达式expr的值为true，则返回v1，否则返回v2。

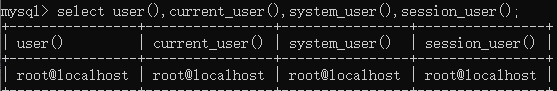
**ifnull(expr,expr2)**：如果表达式expr对应的值不是null，则返回表达式expr的值，否则返回表达式expr2对应的值，若表达式是比较关系，则返回对应的0和1。





**⑵显示当前用户名和主机名**：

**user()、current\_user()、system\_user()、session\_user()**



**⑶得到MySQL当前连接的次数**：**connection\_id()**。

