MySQL中的DQL语句与条件限定where语句

# DQL命令：Data Query Language，数据查询语言

DQL：用来查询**表格中的数据记录**。

关键词 ： **select**



查询数据记录，就是从数据库对象表中获取所要求的数据记录，是数据库中使用频率最高、最重要的数据操作。

# select 语句

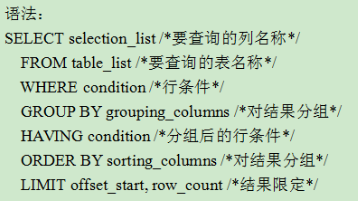
## 基本语法

**select** \* **from** tbName; #查询所有数据记录

\* 表示**所有的字段**，可以查某些特定的字段。

**select** field1,field2,field3,... **from** tbName **where** conditions; #查询限定条件下的数据记录





## 在一个数据库访问另外一个数据库的表格内容

**use** db1; **select** \* **from** tbName; #tbName只能是db1中的表格

**利用数据库名做前缀即可**。 **select \* from db2.tbName;**

## 查询分类

简单查询、条件查询where、范围查询between and、离散查询（in、not in）、去重查询（distinct）、模糊查询（like、not like）、排序查询（order by）、分页查询（limit）、分组查询（group by）、聚合函数（统计查询）（count、**sum、avg、max、min**）。

## 使用示例

**select** \* **from** tableName **where** id = 10；#查询id为10的所有字段值

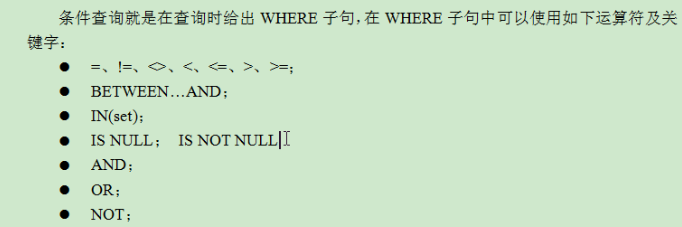
**select** field1 (as) Name,field2 (as) ID **from** tableName **where** conditions; #为字段起别名

查询功能使用较多，单表查询比较简单，多表查询较为复杂。

**select** field1,field2,... **from** tableName **where** id **betweet** 1 **and** 100;#查询id在1到100之间的记录的field1、field2...等字段值。

# where语句：条件查询、更新、删除

这些条件用于构成where语句，where不仅用于查询select语句，也可以适用于update、delete语句。



## 关系运算符：等于、不等于、小于、小于等于、大于、大于等于

=、**!=、<>**、>、>=、<、<=

**where** field **>=** value; 如where score >= 90; 选出大于90分的

等于 只是一个等号=即可；**不等于 新的 != 、旧的 <>**

## BETWEEN …AND… 与 NOT BETWEEN... AND...限定范围

**既可以用于数值区间，也可以指定时间区间。**

注意：前面的数必须小于后面的数。

**使用方法：where** field **between** minValue1 **and** maxValue2;

**where** field **not between** minValue1 **and** maxValue2;

查询某个时间段的数据：

select \* from tbName **where** time **between** ‘2017-01-01 00:00:00’ **and** ‘2018-01-01 00:00:00’;

## IN(…) 与 NOT IN(…)：离散查询

使用方法：**where** field **in**(value1,value2,value3,...);

**where** field **not in**(value1,value2,value3,...);

示例： select \* from citys **where** province **not in**(‘北京’,’天津’,’上海’,’重庆’);

## IS NULL 或者 IS NOT NULL

语句： **where** field **is null**;

**where** field **is not null**;

示例：delete from users where score is null;

## 逻辑运算符AND(&&)、OR(||)、XOR、NOT(!) : 多个条件之间的与或非

使用方法：**where** condition1 **and** condition2 **and**...;

**where** condition1 **or** condition2 **or**...;

**where not** condition1;

**在任何条件前，加上NOT，都是对条件取反。**

## 模糊查询

### 关键字和通配符

关键字：**like 和 not like**

通配符：**%和\_**

**“\_”**只能匹配**单个字符**；**“%”**可以匹配**任意长度**的字符串，可以是0个、1个、多个。

### 语法： where field like ‘\_8’; 或者 where field not like ‘%A%’;

### 模糊查询的功能：

根据字段中具有的某些字符特征进行查询；利用“\_”通配符限制字符个数，因为“\_”只匹配单个字符；可以利用**not like** 找出**不具有某些特征**的数据记录。

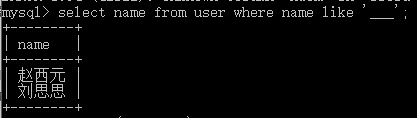
### 示例：

select \* from users\_police **where USERNAME like ‘qinghai%’;** #知道开头

select \* fromusers\_police **where USERNAME like ‘%ai\_po%’**; #知道中间

匹配字符个数：

select \* from user **where name like '\_\_\_**';#三个\_匹配三个字符，一个汉字也算一个字符



### JDBC 中进行模糊查询

需要变量传入拼接时需要使用 $ 连接。

String name = “police”;

String statement = “select \* from users\_police where USERNAME like **‘%$name’**”;

# 字段处理（或控制）查询

## 去重查询：关键字DISTINCT

使用方法：**distinct field**

注意：不加小括号，因为distinct 只是个关键字，不是函数，所以不要distinct(field)。

## 四则运算数据查询

加减乘除和求余：**+-\*/%**(DIV、MOD)

select monthMoney\*12 as yearMoney from money; #通过月工资乘以12获取年薪

## 字段为空(NULL)或查询为空NULL处理

### 有些字段为空，但是需要进行数学运算

注意： **任何数值与NULL相加，结果都是NULL**。

这是需要利用为NULL的返回值0，需要利用IFNULL函数实现：IFNULL(field,0)。

利用**集合函数IFNULL（field，xx）**可以对**NULL**进行过滤，并替换成**新的数据**。

### 根据某些条件，查询不到数据，此时返回为空

select (**select score from user where id = 10**) as score; #返回**NULL**

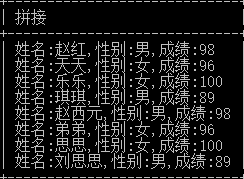
也可以利用IFNULL函数返回特定的值，如‘“不存在”。

select ifnull((select score from user where id = 9),'**不存在**') as score;

## 对查询的数据格式化操作：concat()函数连接字符串

concat()含糊类似format函数，拼接变量和字符串。

select **concat('姓名:',name,',性别:',sex,',成绩:',score)** as '**拼接**' from user;



# 查询某个时间段内的所有记录

## 概述：

MySql的时间字段有**date、time、datetime、timestamp**等，往往我们在存储数据的时候将**整个时间**存在一个字段中，采用**datetime类型**；也可能采用将日期和时间分离，**即一个字段存储date，一个字段存储时间time**。无论怎么存储，在实际应用中，**很可能会出现包含“时间段”类型的查询**，比如一个访问记录数据库，**需要统计每天的访问数量**，这个每天就是一个时间段。下面介绍MySql查询时间段的两种常见方法，其他数据库也可以类似实现。

## 方法一：传统方式：简单，效率低。

即**指定开始时间和结束时间**，用**"between”**或者**"<"，">"、”>=”、“<=”**来建立条件，比如查询2010年3月1日到2010年3月2日的数据条数，则可以使用

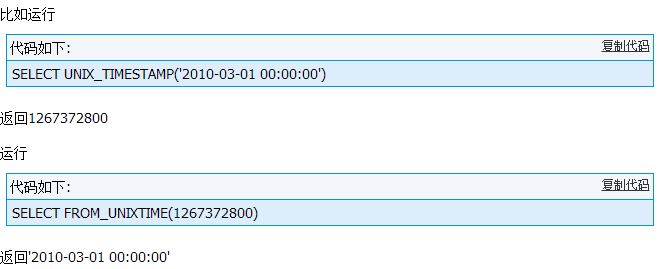
select count(\*) from sometable where datetimecolumn>='2010-03-01 00:00:00' and datetimecolumn<'2010-03-02 00:00:00'

但是，这种方法由于时间不是整数型数据，所以在比较的时候效率较低，所以**如果数据量较大，可以将时间转换为整数型的UNIX时间戳，这就是方法二**。

## 方法二：UNIX时间戳：效率更高

每个时间对应了一个唯一的UNIX时间戳，该时间戳是从'1970-01-01 00:00:00' 为0开始计时，每秒增加1。**MySql内置了传统时间和UNIX时间的互换函数**，分别为：

**UNIX\_TIMESTAMP(datetime)  
FROM\_UNIXTIME(unixtime)**



于是，**我们可以将时间字段里的数据替换为整型的UNIX时间，这样，比较时间就成为整数比较了，建立索引后能大大提高效率。**在查询的时候，需要把起点时间和结尾时间分别转换为UNIX时间再进行比较，如：

select count(\*) from sometable where datetimecolumn>=UNIX\_TIMESTAMP('2010-03-01 00:00:00') and datetimecolumn<UNIX\_TIMESTAMP('2010-03-02 00:00:00')

**也可以在调用程序中先转换为UNIX时间再传入MySql，总之这种方式有利于快速查询时间段，不过显示时间则需要再反转一次**。