# 子查询

## 概述

当一个查询是另一个查询的条件时，称之为子查询。

子查询和连接一样，提供了使用单个查询访问多个表中的数据的方法。子查询可以使用在SELECT、INSERT、UPDATE和DELETE语句中，使它们能够利用子查询返回的结果。

注意：子查询不仅可以使用在WHERE子句中，还经常使用在SELECT子句中。这种情况下，在子查询中要返回的单一值经常是聚合分析的结果。

需要注意的是，子查询必须包含括号。

## 特点

通常来讲，使用子查询的好处如下：

子查询允许结构化的查询，这样就可以把一个查询语句的每个部分隔开。

子查询提供了另一种方法来执行有些需要复杂的JOIN和UNION来实现的操作。

在许多人看来，子查询可读性较高。而实际上，这也是子查询的由来。

## 分类

### 单值子查询

一个子查询如果产生一个单纯的数据，该子查询就如同一个常量，那么就可以像使用常量一样使用它。在实际应用中，我们经常要求子查询只返回一个值，这样就可以将一列值和单个子查询返回值进行比较，这时，可以使用等于（=）、不等于（<>）、大于（>）、小于（<）、大于等于（>=）、小于等于（<=）等运算符。

注：**在SELECT子句中使用子查询时，子查询必须返回单值**。

### 多值子查询

一个子查询除了可以产生一个单一值外，也可以产生一个关系，该关系可以包含若干元组。SQL提供了若干对于关系的操作符，并产生一个布尔型的结果，这些操作符主要用在子查询的结果关系上，它主要包括：IN、EXISTS、SOME（ANY）、ALL、UNIQUE等。

#### IN

#### EXISTS

#### SOME（ANY）

#### ALL

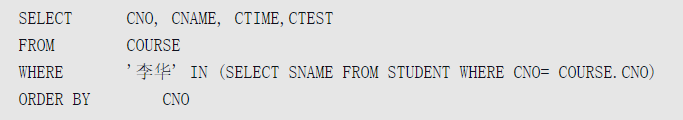
#### UNIQUE

### 独立子查询

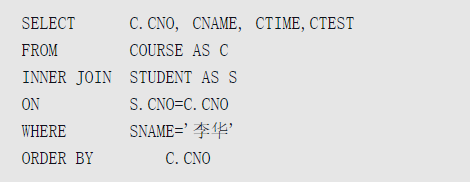
### 相关子查询

#### 使用IN引入相关子查询

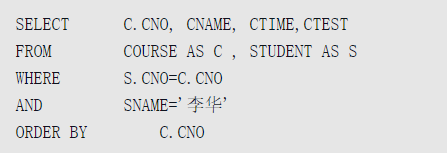
查询COURSE表中李华同学修过的课程的课程号、课程名称、学时和考试时间信息。实例代码如下：



上述代码在执行过程中，DBMS每从COURSE表取出一条记录，都要执行一次子查询。因此，相关子查询具有较高的处理要求，在应用中应尽量避免使用。例如，我们可以采用表的连接实现上面的查询操作，代码如下：

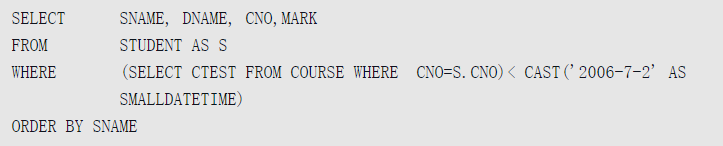


当然，上面表的连接也可以采用带WHERE子句的多表的SELECT语句，代码如下：



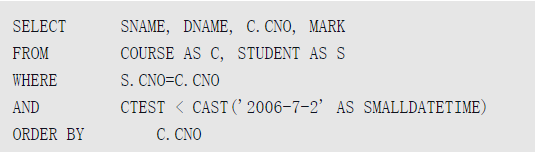
#### 比较运算符引入相关子查询

在STUEDENT表中查询学生的姓名、所在系，课程号及成绩，要求查询的课程的考试时间在2006年7月2日之前。实例代码如下：



比较运算符要求其比较的数据具有相同的数据类型，而STUDENT表中CTEST字段的数据类型为SMALLDATETIME，因此我们需要使用CAST运算式将时间字符串“2006-7-2”转换为SMALLDATETIME型。

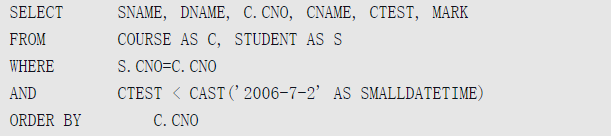
同样，我们可以采用INNER JOIN连接表或者SELECT语句的方法实现上面的查询操作。这里采用带WHERE子句的多表的SELECT语句的实现，代码如下：



可见，二者实现的功能是完全相同的。

与采用相关子查询的方式相比，采用表连接的方式对数据的查询更加灵活。因为，如果采用相关子查询，所能查询的列只能是主查询中FROM子句中的列，不能查询其它列；而采用多表连接的方式，则不存在这个问题，可查询连接的任意表中的列。

在STUEDENT表和COURSE表中查询学生的姓名、所在系，课程号、课程名、考试时间及成绩，要求查询的课程的考试时间在2006年7月2日之前。实例代码如下：

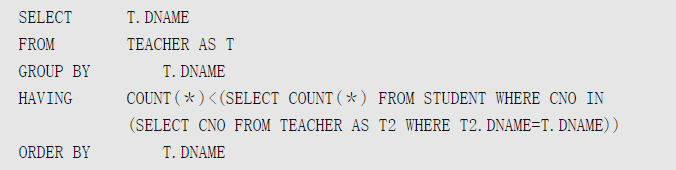


可见，使用多表连接的方式，我们只要在SELECT子句中添加COURSE表中的几个字段，即可得到想要的结果；而如果采用相关子查询的方式，则还需要将COURSE表添加到主查询的FROM子句中，这显然失去了相关子查询的意义。

#### 在HAVING子句中使用相关子查询

在SQL中，DBMS使用WHERE子句中的搜索条件来过滤查询结果表中不想要的行，使用HAVING子句中的搜索条件删除那些不想要的组。

在TEACHER表中查询其包含的系的名称，要求这些系中，各系包含的教师数目要少于STUDENT表中修过该系的教师开设课程的学生的人次。假如TEACHER表中，A系有m个教师，而在STUDENT表中有n人次修过这m个教师开设的课程，如果n>m，则A系就符合查询条件。实例代码如下：



### 嵌套子查询

大多数情况下，子查询出现在查询的HAVING子句或WHERE子句中。子查询自身也可以包含WHERE子句和或HAVING子句，同样，子查询也可以出现在其他子查询中。位于其他子查询内的子查询被称为嵌套的子查询。

### 树查询

## 使用场景

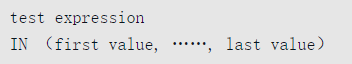
使用子查询可以避免由于子查询中的数据产生的重复

使用子查询更符合语意，更好理解

# 原理

## IN子查询

IN运算符允许根据一行记录中是否有一列包括在一系列值之中而选择该行。语法如下。



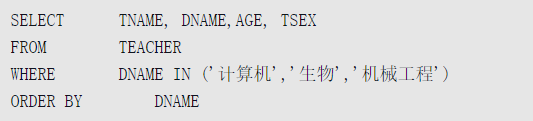
所有的条件在IN运算符后面罗列，并以括号“()”包括起来，条件中间用逗号分开。当testexpression处于括号中列出的一系列值之中时，IN运算符求值为True。

与IN运算符相对的还有NOT IN运算符，其判别结果刚好与IN运算符相反，即当test expression不处于括号中列出的一系列值之中时，NOT IN运算符求值为True。

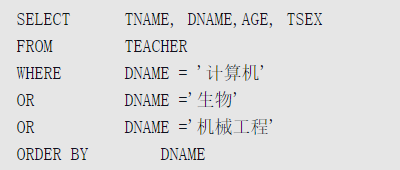
### IN子查询与OR运算符

**在大多数情况下，OR运算符与IN运算符可以实现相同的功能**。

例如：



等价于：



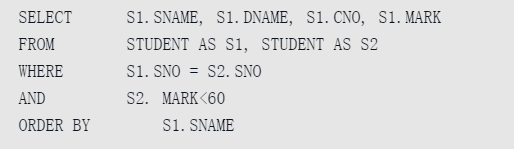
我们可以发现二者结果完全相同。那么为什么要提出IN运算符呢？这是因为与OR运算符相比，IN运算符具有以下两个优点。

• 当选择的条件很多时，采用IN运算符就显得很简便，只需在括号内用逗号间隔依次罗列即可，运行效率比OR运算符要高。

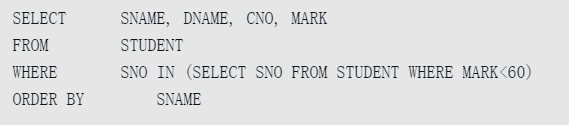
• 使用IN运算符，其后面所列的条件可以是另一条SELECT语句，即子查询。

### IN子查询与自连接

可以使用IN实现自连接：



采用IN子查询的方式如下：



分析一下代码的执行过程。

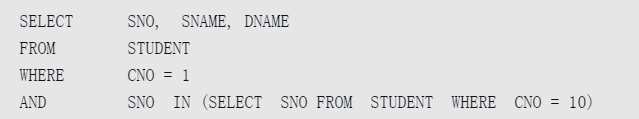
1. 执行一个查询时，DBMS首先处理最里面的子查询。系统首先执行“SELECT SNO FROMSTUDENT WHERE MARK<60”语句，生成STUDENT表中成绩小于60的学生的学号SNO集。
2. DBMS一次处理STUDENT表中的一行记录，并且将每行记录中的SNO列值与子查询结果集中的SNO值进行比较。
3. 如果在系统子查询结果集中找到了与正在处理的记录中的SNO值相匹配的值，WHERE子句求值为True，DBMS则将该记录的相关信息归入结果表。
4. 如果在子查询结果集中没有发现与记录的SNO值相匹配的值，WHERE子句求值为False，DBMS转而去处理STUDENT表中的下一行而舍弃该行数据。

### IN子查询与集合交/差集

在STUDENT表中查询修过1号或者10号课程的学生的学号、姓名及所在系信息。

分析：该实例实际上可理解为修过1号课程的学生的相关信息与修过10号课程的学生的相关信息的交集。

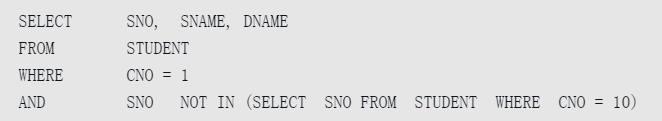
实例代码如下：



在STUDENT表中查询修过1号课程但是没有修10号课程的学生的学号、姓名及所在系信息。

分析：该实例实际上可理解为修过1号课程的学生的相关信息与修过10号课程的学生的相关信息的差集。

实例代码如下：



注意：由运算符IN引入的子查询返回的列值，既可以来自主查询的表，也可以来自其他表。SQL对查询的唯一要求就是它必须返回单一列的数据值，并且其数据类型必须与IN前面表达式的数据类型相兼容。

## EXIST子查询

**IN和EXIST区别：**

mysql中的in语句是把外表和内表作hash 连接，而exists语句是对外表作loop循环，每次loop循环再对内表进行查询。一直大家都认为exists比in语句的效率要高，这种说法其实是不准确的。这个是要区分环境的。

1、如果查询的两个表大小相当，那么用in和exists差别不大。

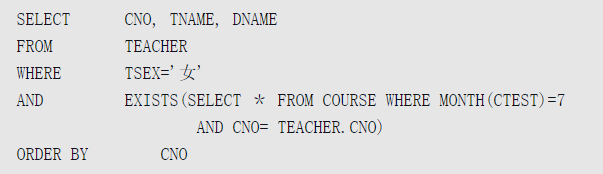
2、如果两个表中一个较小，一个是大表，则子查询表大的用exists，子查询表小的用in。

3、not in 和not exists如果查询语句使用了not in 那么内外表都进行全表扫描，没有用到索引；而not extsts的子查询依然能用到表上的索引。所以无论那个表大，用not exists都比not in要快。

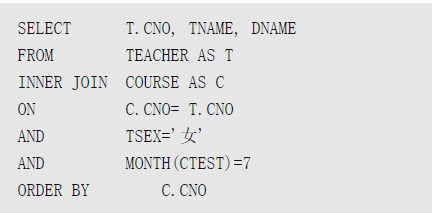
### EXISTS子查询与交集

在TEACHER表中查询女教师开设的、6月份考试的课程号、开课教师的姓名以及教师所在的系。分析：实际上我们要查询的课程为TEACHER表中女教师开设的课程与COURSE表中7月考试的课程的交集。

实例代码如下：



也可以采用INNER JOIN方式，代码如下：



## NOT运算符

NOT运算符用于对搜索条件的布尔值求反。本节将讨论NOT运算符的使用、应用场合及其与<>运算符的区别。

注：**NULL值进行取反，结果仍是NULL**。

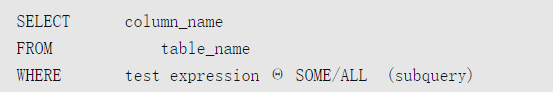
对于简单的条件查询，NOT运算符与<>运算符的功能几乎没有什么区别，那么NOT运算符的优势体现在哪里呢？答案是它可以与其他运算符组合使用，这一点是<>运算符所不能实现的。

注：MySQL数据库系统不支持NOT运算符。在MySQL数据库系统中，NOT运算符只能被用在EXISTS运算符的前面，形如NOT EXISTS。

## SOME/ANY/ALL子查询

只要我们使用了SQL比较运算符（等于=、不等于<>、大于>、小于<、大于等于>=、小于等于<=）中的一个来比较两个表达式的值，那么运算符前后的表达式都必须为单一值。正如10.1节所介绍的那样，只有当子查询返回值为单值时，才可使用子查询作为比较判式的表达式之一。

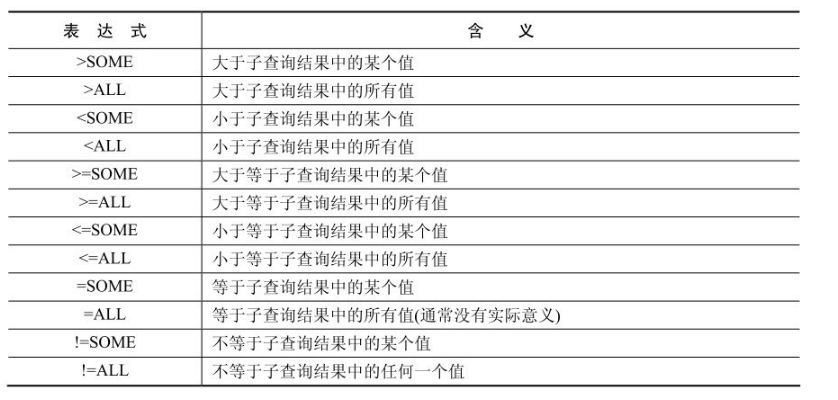
而数量词SOME、ANY和ALL则允许使用比较运算符将单值与子查询返回的值加以比较，这里的子查询返回的结果可以是多行的。语法如下：



test expression可以是实际值、列名、表达式或是另一个返回单一值的子查询。Θ为比较运算符，可以是6种SQL比较运算符中的任意一种。我们如果采用SOME关键字，则表示表达式的值至少与子查询结果中的一个值相比，满足比较运算符Θ；而如果采用ALL关键字，则表示表达式的值与子查询结果中所有的值相比，都满足比较运算符Θ。

**注：**关键字SOME和关键字ANY具有完全相同的功能。

SOME/ALL关键字与比较运算符的联合使用：



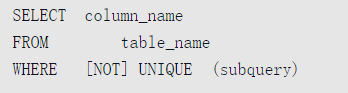
SOME和ALL与聚集函数的对应关系：



## UNIQUE

UNIQUE运算符用来测试集合是否存在重复元组。与EXISTS判式相似，它总是与子查询结合使用，而且只要子查询结果中没有重复记录，UNIQUE判式的值就为True；如果子查询的结果表中有重复的记录，那么UNIQUE判式的值为False。

简单的UNIQUE子查询语法可表示如下：



这里需要说明的是，只要子查询subquery中返回的结果表存在重复记录，则UNIQUE (subquery)就返回True，WHERE子句的结果也就为True；当然，如果使用NOT UNIQUE判式，当子查询返回的结果有重复的记录行时，它返回True。

要使UNIQUE判式有意义，我们应该在子查询中建立搜索条件，以匹配子查询连接起来的两个表中的值。

# 应用

## 多例过滤

子查询匹配两个值，机MySQL独有的多例过滤方式：

SELECT a.user\_name,b.timestr,kills

FROM user1 a

JOIN user\_kills b ON a.id = b.user\_id

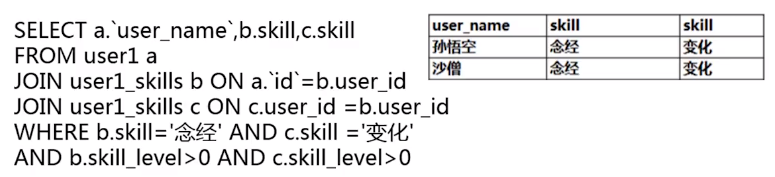
WHERE (b.user\_id,b.kills) IN (

SELECT user\_id,MAX(kills) FROM user\_kills GROUP BY user\_id

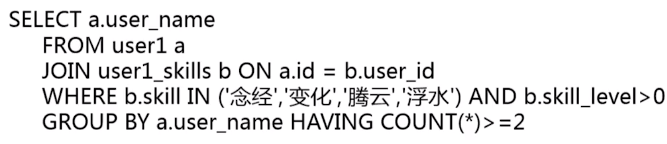
);

## 同属性多值过滤

### JOIN方式实现



### GROUP BY方式实现



## 多属性查询