昨日回顾 2

查询语句 3

基本查询 3

orderby子句 3

limit子句 3

连接查询 4

基本含义 4

连接语法的基本形式 5

交叉连接： 5

内连接： 6

左［外］连接： 9

右［外］连接： 10

全［外］连接： 11

连接查询举例： 11

子查询 12

什么叫子查询 12

子查询按结果分类： 13

按使用场合分： 13

常见子查询及相关关键字 14

使用in子查询 15

使用any子查询 15

使用ａｌｌ子查询 16

使用some的子查询 17

使用exists的子查询 17

使用not exists子查询 18

联合查询 18

基本含义 19

语法形式： 19

昨日回顾

[数据操作语言（ＤＭＬ）](#Toc23380)：

[数据插入（添加数据）](#Toc608)

[有3种形式](#Toc32449)：

**insert into 表名(字段名1，字段名2，.....) values (值1，值2，......)， (值1，值2，......)，.....**

insert into 表名(字段名1，字段名2，.....) select 字段名1，字段名2，..... from 表明2 .....；

insert into 表名 set 字段名1=值1，字段名2=值2， ....;

[其他类似插入数据的语句：](#Toc31812)

复制表方法1：

第一步：create table tab2 like tab1；

第二步：insert into tab2 select \* from tab1;

复制表方法2（不推荐）：

create table tab2 select \* from tab1；

导入“结构整齐”的数据：

load data infile 文件完整路径 into table 表名；

[删除数据](#Toc1764)

[基本语法形式：](#Toc10877)delete from 表名 [where子句] [order by子句] [limit子句]；

假删：只要在数据表的字段上设定一个字段，用于表示“是否删除”的意思（boo类型）

删除： update tab1 set deleted = 1;

取得： select \* from tab1 where **deleted=0**;

[类似删除语句truncate：](#Toc18595)

truncate [table] 表名 ；

[修改数据](#Toc25595)

[基本语法：](#Toc32732)

update 表名 set 字段名1=值1，字段名2=值2， ....; [where子句]

[数据查询语言DQL](#Toc5226)

[基本查询](#Toc11170)

[语法形式](#Toc4775)：

select [ALL |DISTINCT]结果项列表 from子句 where子句 group子句 having子句 order子句limit子句

[解释说明：](#Toc32062)

上述语法结构中，每一项的“相对先后顺序”不能打乱。实际上，在其内部计算的过程中，也是按此先后顺序进行的。

[[all | distinct]](#Toc11637)：显示全部（不消除重复项)（默认）,或消除重复项

[from子句](#Toc8207)：数据来源，就是表，也可以多个表的某种结合关系。

[where子句](#Toc1266)：对前述数据来源设定的过滤条件。

[group by 分组子句](#Toc3295)：对前述过滤之后的数据进行分组。分组之后的结果就只有“组信息”，常见的有如下几个：

1，分组依据本身的信息。

2，组的个数（记录数）

3，组中某些字段（数值字段）的最大值，最小值，总和值，平均值

[having子句](#Toc19270)：

对前述已经分组的数据的组设定的过滤条件

查询语句

基本查询

orderby子句

形式：

order by 排序字段1 [排序方式]， 排序字段2 [排序方式]， .....

说明：

对前面取得的数据（含from子句，where子句，group子句，having子句的所有结果）来指定按某个字段的大小进行排列（排序），排序只有2种方式：

正序： ASC（默认值），可以省略

倒序： DESC

如果指定多个字段排序（虽然不常见），则其含义是，在前一个字段排序中相同的那些数据里，再按后一字段的大小进行指定的排序。





limit子句

形式：

limit [起始行号start]， 要取出的行数num

说明：

表示将前面取得的数据并前面排好之后（如果有），对之指定取得“局部连续的若干条”数据。

起始行号ｓｔａｒｔ：第一行的行号为０，　可以省略，则为默认行号（０）。

要取得的行数：如果结果集中从指定的行号开始到最后没有这么多行，则就只取到最后。

此子句非常有用——主要用于网页上最常见的一个需求（现象）：分页。

分页原理：

分页的前提：人为指定每页显示的条数，$pageSize = 3;

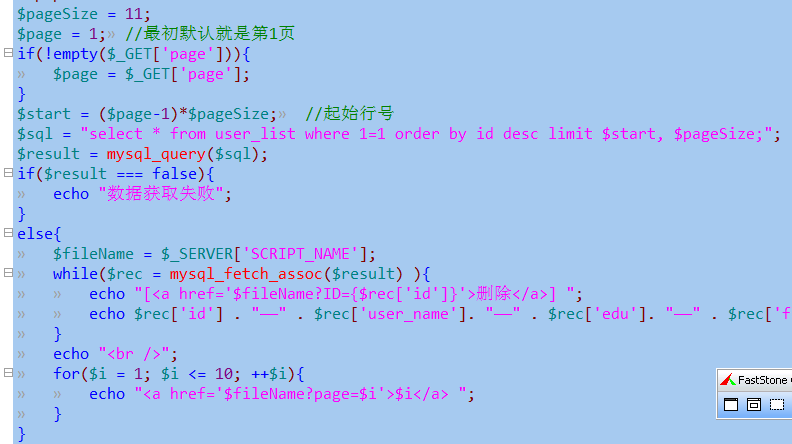
显示（取得）第1页数据：select \* from 表名 limit 0, $pageSize;

显示（取得）第2页数据：select \* from 表名 limit 3, $pageSize;

显示（取得）第3页数据：select \* from 表名 limit 6, $pageSize;

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

显示（取得）第$n页数据：select \* from 表名 limit ($n-1)\*$pageSize, $pageSize;



连接查询

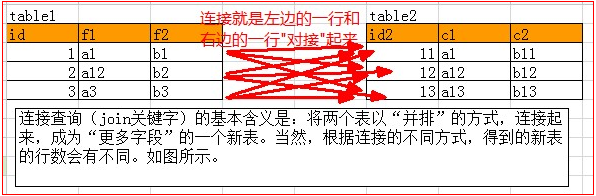
基本含义

连接就是指两个或2个以上的表（数据源）“连接起来成为一个数据源”。

实际上，两个表的完全的连接是这样的一个过程：

左边的表的每一行，跟右边的表的每一行，两两互相“横向对接”后所得到的所有数据行的结果。

注意：连接之后，并非形成了一个新的数据表，而只是一种“内存形态”。



连接语法的基本形式

from 表1 [连接方式] join 表2 [on 连接条件]；

连接的结果可以当作一个“表”来使用。常用有以下几种连接方式：

交叉连接：

实际上，交叉连接是将两个表不设定任何条件的连接结果。

交叉连接通常也被叫做“笛卡尔积”——数学上可能比较多。

语法：

from 表1 [cross] join 表2 ; //可见交叉连接只是没有on条件而已。

cross这个词也可以省略，还可以使用inner这个词代替

表1：

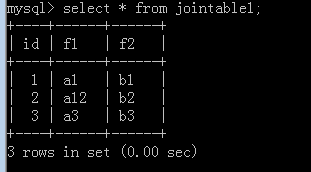
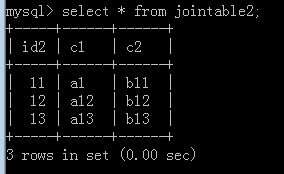
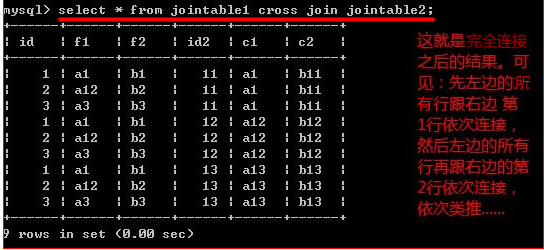


表2：



交叉：

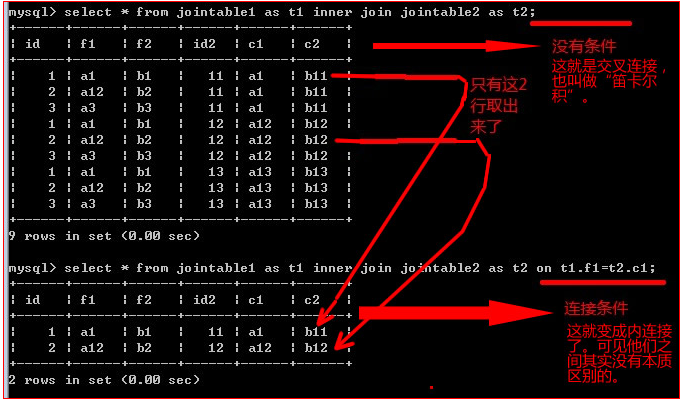


内连接：

语法：

from 表1 [inner] join 表2 on 表1.字段1=表2.字段2；

含义：找出（过滤）在交叉连接的结果表中的表1的字段1的值等于表2的字段2的值的那些行。



分析交叉连接的“无意义性”，以及内连接的“有意义性”：

先看交叉连接：





再看内连接：





可见，在现实的常见需求中，上述内连接的结果就都是“有意义”的数据了。

也可以指定只列出其中部分字段：

[SELECT](http://www.phpmyadmin.com/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html&server=0&token=9ca89acccb7223501d39e69b99f18433) **t1.pro\_id, pro\_name, price, t2.protype\_name** FROM product as t1 join `product\_type` as t2 on t1.protype\_id=t2.protype\_id WHERE 1

结果为：



左［外］连接：

形式：

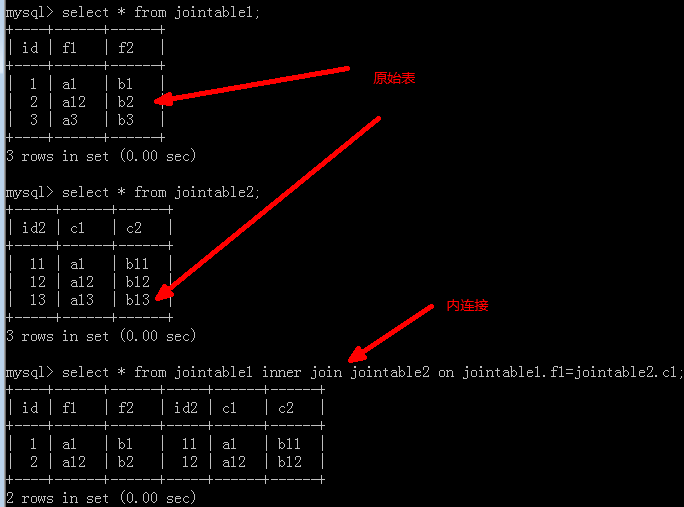
from 表1 left [outer] join 表2 on 连接条件。

说明：

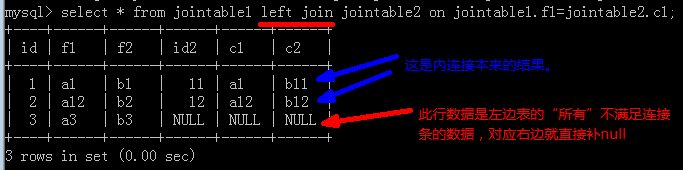
1，这里，left是关键字。

2，连接条件跟内连接一样。

3，含义是：内连接的结果基础上，加上左边表中所有不符合连接条件的数据，相应本应放右边表的字段的位置就自动补为“null”值。



则左连接结果为：



右［外］连接：

右连接跟左连接恰恰相反：

形式：

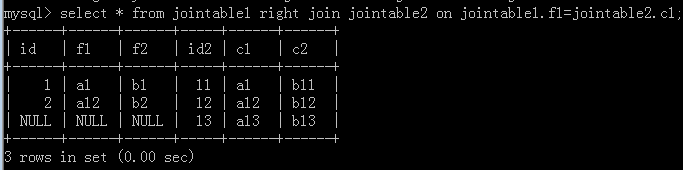
from 表1 right [outer] join 表2 on 连接条件。

说明：

1，这里，right是关键字。

2，连接条件跟内连接一样。

3，含义是：在内连接的结果基础上，加上右边表中所有不符合连接条件的数据，相应本应放左边表的字段的位置就自动补为“null”值。



全［外］连接：

形式：

from 表1 full [outer] join 表2 on 连接条件；

说明：

1，含义：其实是左右连接的“并集”（消除重复项），即内连接的结果，加上左表中不满足条件的所有行（右边对应补null），再加上，右表中不满足条件的所有行（左边对应补null）。

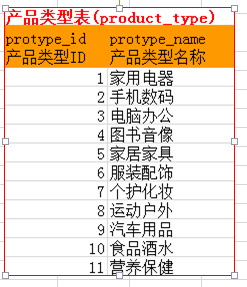
2，mysql中其实不认识全[外]连接语法，即mysql这个软件本身不支持全连接的语法。

3，此概念在其他数据库有的存在，了解就可以。

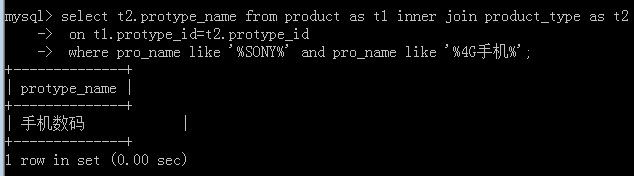
连接查询举例：

原始数据：

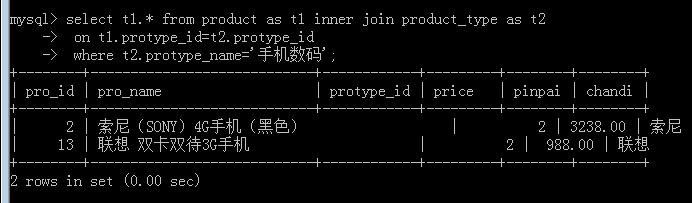




1：找出索尼4G手机所属类别名称：



2，找出所有属于手机数码的产品：



子查询

什么叫子查询

一个查询，通常就是一个select语句（即出现一次select关键字）

但，如果在一个select查询语句中，又出现了select查询语句，此时就称后者为“子查询”，前者就是“主查询”

形式：

* **selelct 字段或表达式或(子查询1) [as 别名] from 表名或(子查询2) where 字段或表达式或(子查询3) 的条件判断**

注意：

每个位置所放置的子查询结果，应该符合该位置的数据需求。

通常：

子查询1应该是一个“数据结果”。

子查询2可以是“任意结果”，此位置的查询结果，通常作为数据源**，**必须**给一个别名**

子查询3可以是一个数据或一列数据甚至是一行数据

子查询按结果分类：

* 表子查询 ： 一个子查询返回的结果理论上是“多行多列”的时候。此时可以当做一个“表”来使用，通常是放在from后面。
* 行字查询 ： 一个子查询返回的结果理论上是“一行多列”的时候。此时可以当做一个“行”来使用，通常放在“行比较语法”中。
* 列子查询 ： 一个子查询返回的结果理论上是“多行一列”的时候。此时可以当做“多个值”使用，类似这种：(5, 17, 8, 22)。
* 标量子查询：一个子查询返回的结果理论上是“一行一列”的时候。此时可以当做“一个值”使用，类似这种：select 5 as c1; 或select ...where a = 17，或select ... where b > 8;

按使用场合分：

* 作为主查询的结果数据：select c1,**(select f1 from tab2) as f11** from tab1; #这里子查询应该只有一个数据（一行一列，标量子查询）
* 作为主查询的条件数据：select c1 from tab1 where c1 in **(select f1 from tab2)**; #这里子查询可以是多个数据（多行一列，列子查询，以及标量子查询，实际上行子查询也可能，但极少）
* 作为主查询的来源数据：select c1 from **(select f1 as c1, f2 from tab2) as t2**; #这里子查询可以是任意查询结果（表子查询）。

常见子查询及相关关键字

* 比较运算符中使用子查询

形式为： **操作数 比较运算符 (标量子查询)**；

操作数通常是一个字段。

含义：判断该操作数（字段）的值是否满足该比较运算符所设定的比较结果。

其实就是相当于最简单的这种形式：　id > 5;

举例：

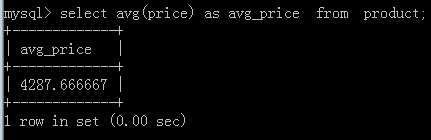
数据源：



需求：找出所有大于平均价的商品。

第一步：找平均价：

select avg(price) as avg\_price from product;



第二部：找商品：

select \* from product where price > 4287.7; ==〉

select \* from product where price 〉（select avg(price) as avg\_price from product）；



使用in子查询

in的基本语法形式为：

where 操作数 in (值1，值2， ....）

则in子查询就是：

where 操作数 in ( 列子查询 ）；

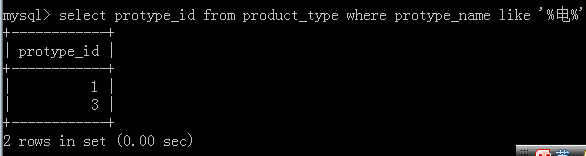
含义：

表示该操作数（字段值） 等于 该子查询的其中任意一个只，就算满足条件。

举例：

找出所有带“电”字的类别的产品

第一步：找出所有带“电”字的类别ID：



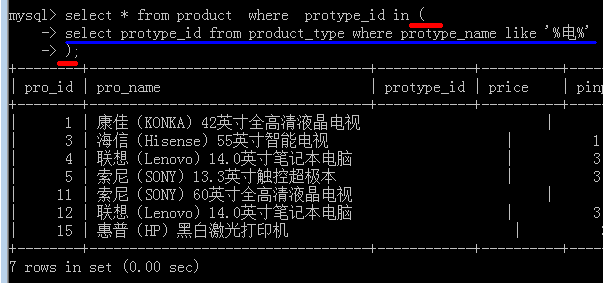
第二步：根据结果找出这些类别的产品：

select \* from product where protype\_id in (1, 3); ==〉

select \* from product where protype\_id in (

select protype\_id from product\_type where protype\_name like '%电%'

);



使用any子查询

使用形式：

where 操作数 比较运算符 any ( 列子查询 );

说明：

１操作数通常仍然是字段名

２比较运算符就是常规的〉　　〉＝　　＜　　　＜＝　　＝　　　＜＞

３列子查询也可以是标量子查询，都表示“若干个数据值”

含义：

表示该操作数的值只要跟列子查询的任意一个值满足给定的比较运算，就算满足了条件——就是只要有一个成就成。

考察一个特定情况：

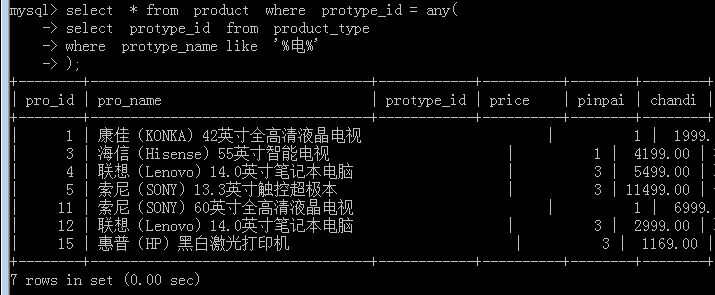
where 操作数 = any ( 列子查询 );

则其完全相当于：

where 操作数 in ( 列子查询 );

举例：

找出所有带“电”字的类别的产品



使用ａｌｌ子查询

where 操作数 比较运算符 all ( 列子查询 );

说明：

１操作数通常仍然是字段名

２比较运算符就是常规的〉　　〉＝　　＜　　　＜＝　　＝　　　＜＞

３列子查询也可以是标量子查询，都表示“若干个数据值”

含义：

表示该操作数的值必须跟列子查询的所有值都满足给定的比较运算，才算满足了条件。

举例：

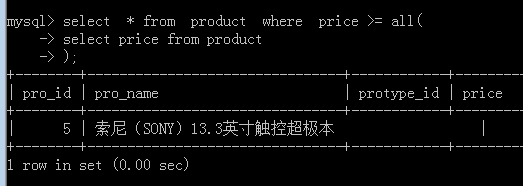
找出产品表中的价格最高的产品。

分析：最高价的产品的价格会大于等于所有产品价格。

select \* from product where price >= all(

select price from product

);

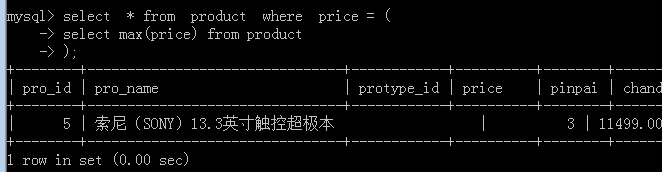


方法二：

select \* from product where price = (

select max(price) from product

);



使用some的子查询

some是any的同义词。一样用。

使用exists的子查询

以前所学：

if exist

或者：

if not exists

以前其含义通常是：存在某数据（通常是一个表）

现在，该词也同样用于子查询，表示同样的含义“存在”

形式：

where exists （子查询);

含义：

如果该子查询有结果数据（无论什么数据，只要大于等于１行），则就是true，否则为false

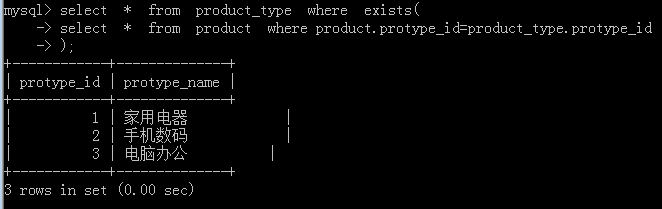
举例：

找出具有在售商品的那些类别：

select \* from product\_type where exists(

select \* from product where product.protype\_id=product\_type.protype\_id

);



使用not exists子查询

含义跟exists子查询相反。

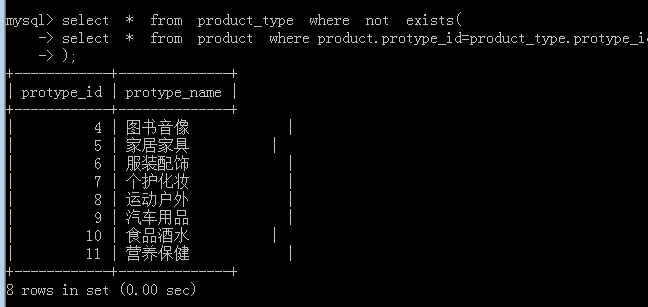
举例：

找出还没有在售商品的类别：

select \* from product\_type where not exists(

select \* from product where product.protype\_id=product\_type.protype\_id

);



了解一下：实际上，这种exists（或not exists）子查询，如果涉及到2个表（或以上），其内部其实会自动进行“连接查询”，且其逻辑过程较为负责，而且还不明确，通常认为属于效率较低的子查询，尽量少用。

联合查询

联合查询的关键字是： union

连接查询的关键字是： join

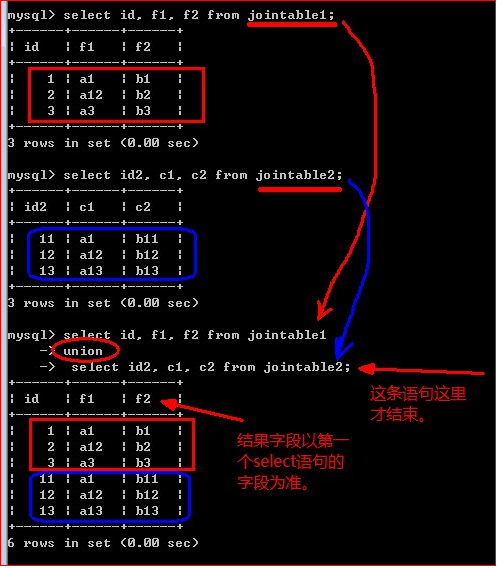
但，在mysql手册中， join这个连接查询，往往都翻译为“联合查询”

但在绝大多数的中文书籍和文章中，join被翻译为“连接查询”

基本含义

联合查询就是将两个select语句的查询结果“层叠”到一起成为一个“大结果”。

两个查询结果的能够进行“联合”的先觉条件是：结果字段数相等。



语法形式：

select 语句1

union [ALL | DISTINCT]

select 语句2；

说明：

1，两个select语句的输出段（结果字段）一样数目一样，应用中通常类型一样才有意义。

2，结果集中的字段以第一个select语句的字段为准。

3，第一个select语句的字段可以做别名，但如果做别名，则后续的where，group，order等子句应该用该别名。

4，联合查询默认是会消除重复项的（DISTINCT），要想不消除，则必须明确些“ALL”。

5，如果要对整个联合结果进行排序或limit，则应该对各自的select语句加括号：

（select 语句1）

union

（select 语句2）

order by ..... limit ....；

