# 业务数据查询-1

## 单表查询

### 数据准备

自来水收费系统建表语句.sql

### 简单条件查询

#### 精确查询

需求 ：查询水表编号为 30408 的业主记录

查询语句 ：

|  |
| --- |
| select \* from t\_owners where watermeter= '30408'; |

查询结果 ：



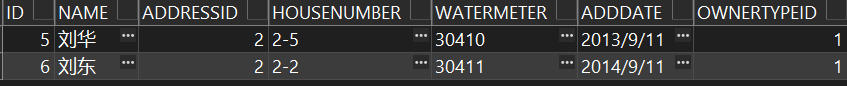
#### 模糊查询

需求 ：查询业主名称包含“刘”的业主记录

查询语句 ：

|  |
| --- |
| select \* from t\_owners where name like '%刘%' |

查询结果 ：



#### and 运算符

需求 ：查询业主名称包含“刘”的并且门牌号包含 5 的业主记录

查询语句 ：

|  |
| --- |
| select \* from t\_owners where name like '%刘%' and housenumber like '%5%'; |

查询结果 ：



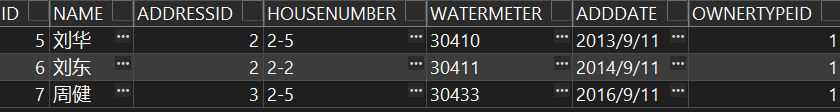
#### or 运算符

需求 ：查询业主名称包含“刘”的或者门牌号包含 5 的业主记录

查询语句 ：

|  |
| --- |
| select \* from t\_owners  where name like '%刘%' or housenumber like '%5%'; |

查询结果 ：



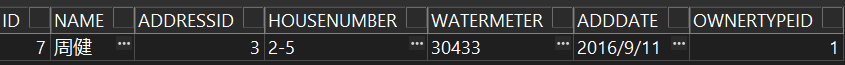
#### and 与 or 运算符混合使用

需求 ：查询业主名称包含“刘”的或者门牌号包含 5 的业主记录 ，并且地址编号 为 3 的记录。

语句 ：

|  |
| --- |
| select \* from t\_owners  where (name like '%刘%' or housenumber like '%5%')  and addressid=3 |

查询结果 ：



因为 and 的优先级比 or 大 ，所以我们需要用 ( ) 来改变优先级。

#### 范围查询

需求 ：查询台账记录中用水字数大于等于 10000 ，并且小于等于 20000 的记录

我们可以用>= 和<=来实现

|  |
| --- |
| select \* from t\_account  where usenum>=10000 and usenum<=20000; |

我们也可以用 between .. and ..来实现

|  |
| --- |
| select \* from t\_account  where usenum between 10000 and 20000 |

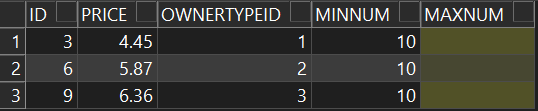
#### 空值查询

需求 ：查询 T\_PRICETABLE 表中 MAXNUM 为空的记录

语句 ：

|  |
| --- |
| select \* from t\_pricetable t where maxnum is null |

查询结果 ：

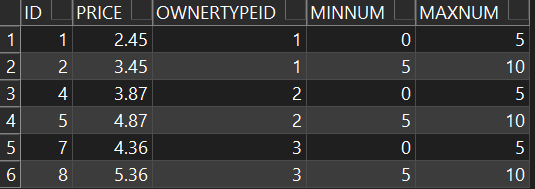


需求 ：查询 T\_PRICETABLE 表中 MAXNUM 不为空的记录

语句 ：

|  |
| --- |
| select \* from t\_pricetable t where maxnum is not null |

查询结果 ：



### 去掉重复记录

需求 ：查询业主表中的地址 ID,不重复显示

语句 ：

|  |
| --- |
| select distinct addressid from t\_owners; |

### 排序查询

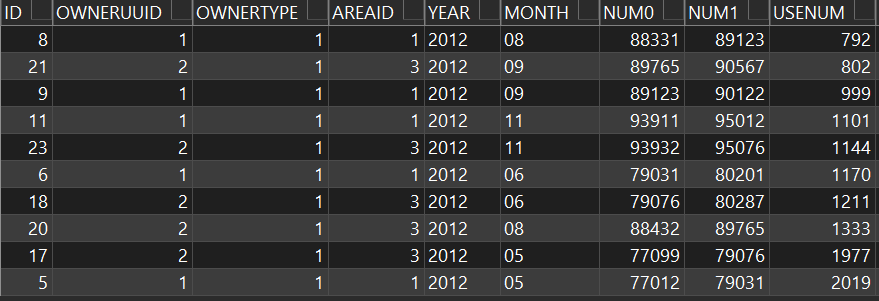
1.升序排序

需求 ：对 T\_ACCOUNT 表按使用量进行升序排序

语句 ：

|  |
| --- |
| select \* from t\_account order by usenum; |

查询结果 ：



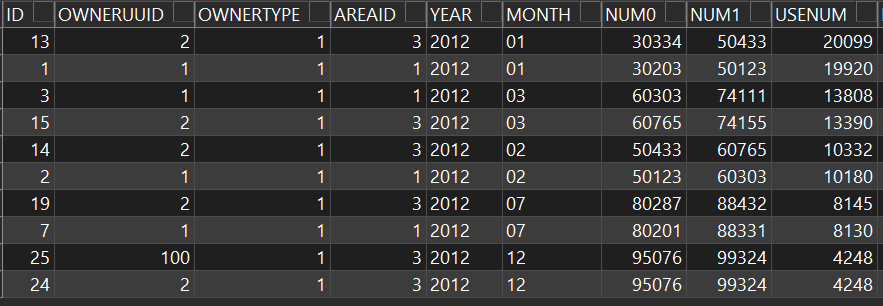
2.降序排序

需求 ：对 T\_ACCOUNT 表按使用量进行降序排序

语句 ：

|  |
| --- |
| select \* from t\_account order by usenum desc; |

查询结果 ：



### 基于伪列的查询

在 Oracle 的表的使用过程中 ，实际表中还有一些附加的列，称为伪列。

伪列就 像表中的列一样 ，但是在表中并不存储。伪列只能查询，不能进行增删改操作。

接下来学习两个伪列 ：ROWID 和 ROWNUM。

1 ROWID

表中的每一行在数据文件中都有一个物理地址 ，ROWID 伪列返回的就是该行的 物理地址。

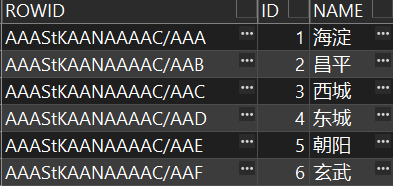
使用 ROWID 可以快速的定位表中的某一行。ROWID 值可以唯一的标识表中的一行。

由于 ROWID 返回的是该行的物理地址 ，因此使用 ROWID 可以显示行是如何存储的。

查询语句：

|  |
| --- |
| select rowid, t.\* from t\_area t |

查询结果如下：



我们可以通过指定 ROWID 来查询记录

|  |
| --- |
| select rowID,t.\*  from T\_AREA t  where ROWID= 'AAAStKAANAAAAC/AAC'; |

查询结果如下 ：



2 ROWNUM

在查询的结果集中 ，ROWNUM 为结果集中每一行标识一个行号 ，第一行返回 1 ， 第二行返回 2 ，以此类推。通过 ROWNUM 伪列可以限制查询结果集中返回的行数

查询语句:

|  |
| --- |
| select rownum,t.\* from t\_owners t |

查询结果如下 ：



我们的分页查询需要用到此伪列，在本章第四小节详细讲解。

### 聚合统计

ORACLE 的聚合统计是通过分组函数来实现的 ，与 MYSQL 一致。

1. 聚合函数

( 1 ) 求和 sum

需求 ：统计 2012 年所有用户的用水量总和

|  |
| --- |
| select sum(usenum) from t\_account where year= '2012' |

查询结果如下 ：



( 2 ) 求平均 avg

需求 ：统计 2012 年所有用水量 (字数) 的平均值

|  |
| --- |
| select avg(usenum) from T\_ACCOUNT where year= '2012' |

查询结果如下 ：



( 3 ) 求最大值 max

需求 ：统计 2012 年最高用水量 (字数)

|  |
| --- |
| select max(usenum) from T\_ACCOUNT where year= '2012' |

查询结果如下 ：



( 4 ) 求最小值 min

需求 ：统计 2012 年最低用水量 (字数)

|  |
| --- |
| select min(usenum) from t\_account where year= '2012' |

查询结果如下 ：

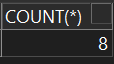


( 5 ) 统计记录个数 count

需求 ：统计业主类型 ID 为 1 的业主数量

|  |
| --- |
| select count(\*) from t\_owners t where ownertypeid=1 |

查询结果如下 ：



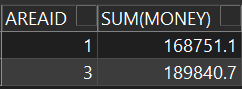
2. 分组聚合 Group by

需求 ：按区域分组统计水费合计数

语句 ：

|  |
| --- |
| select areaid, sum(money) from t\_account group by areaid |

查询结果 ：



3. 分组后条件查询 having

需求 ：查询水费合计大于 16900 的区域及水费合计

语句 ：

|  |
| --- |
| select areaid,sum(money) from t\_account  group by areaid  having sum(money)>169000 |

查询结果 ：



## 多表查询

### 多表内连接查询

( 1 ) 需求 ：查询显示业主编号 ，业主名称 ，业主类型名称

如下图 ：



查询语句 ：

|  |
| --- |
| select  t1.id, t1.name, t2.name  from t\_owners t1, t\_ownertype t2  where t1.ownertypeid=t2.id  ; |

|  |
| --- |
| select  t1.id, t1.name, t2.name  from t\_owners t1  inner join t\_ownertype t2 on t1.ownertypeid=t2.id  ; |

( 2 ) 需求 ：查询显示业主编号 ，业主名称、地址和业主类型 ，如下图



分析 ：此查询需要三表关联查询。分别是业主表 ，业主分类表和地址表

语句 ：

|  |
| --- |
| select o.id 业主编号, o.name 业主名称, ad.name 地址, ot.name 业主类型  from t\_owners o,t\_ownertype ot,t\_address ad  where o.ownertypeid=ot.id and o.addressid=ad.id |

|  |
| --- |
| select  t1.id 业主编号,  t1.name 业主名称,  t2.name 地址,  t3.name 业主类型  from t\_owners t1  join t\_address t2 on t1.addressid=t2.id  join t\_ownertype t3 on t1.ownertypeid=t3.id  ; |

( 3 ) 需求 ：查询显示业主编号、业主名称、地址、所属区域、业主分类 ，如下图 ：



分析 ：这里需要四个表关联查询 ，比上边多了一个区域表 ( T\_AREA )

查询语句 ：

|  |
| --- |
| select o.id 业主编号,o.name 业主名称,ar.name 区域,  ad.name 地址, ot.name 业主类型  from t\_owners o ,t\_ownertype ot,t\_address ad,t\_area ar  where o.ownertypeid=ot.id and o.addressid=ad.id  and ad.areaid=ar.id |

( 4 ) 需求 ：查询显示业主编号、业主名称、地址、所属区域、 收费员、业主分 类 ，如下图 ：



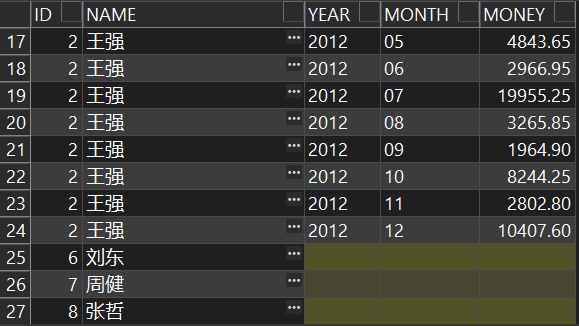
分析 ：此查询比上边又多了一个表 T\_OPERATOR

语句 ：

|  |
| --- |
| select ow.id 业主编号, ow.name 业主名称, ad.name 地址,  ar.name 所属区域, op.name 收费员, ot.name 业主类型  from t\_owners ow,t\_ownertype ot,t\_address ad, t\_area ar, t\_operator op  where ow.ownertypeid=ot.id and ow.addressid=ad.id  and ad.areaid=ar.id and ad.operatorid=op.id |

### 左外连接查询

需求 ：查询业主的账务记录 ，显示业主编号、名称、年、月、金额。如果此业主 没有账务记录也要列出姓名。



分析 ：我们要查询这个结果 ，需要用到 T\_OWNERS (业主表) ，T\_ACCOUNT (台账表) 按照查询结果 ，业主表为左表、账务表为右表。

按照 SQL1999 标准的语法 ，查询语句如下 ：

|  |
| --- |
| SELECT ow.id, ow.name, ac.year, ac.month, ac.money  FROM t\_owners ow left join t\_account ac on ow.id=ac.owneruuid |

按照 ORACLE 提供的语法 ，就很简单了 ：

|  |
| --- |
| SELECT ow.id, ow.name, ac.year, ac.month,ac.money  FROM t\_owners ow,t\_account ac  WHERE ow.id=ac.owneruuid(+) |

如果是左外连接 ，就在右表所在的条件一端填上 (+)

### 右外连接查询

需求 ：查询业主的账务记录 ，显示业主编号、名称、年、月、金额。

如果账务记录没有对应的业主信息 ，也要列出记录。如下图 ：



SQL1999 标准的语句

|  |
| --- |
| select ow.id, ow.name, ac.year, ac.month, ac.money  from t\_owners ow right join t\_account ac on ow .id=ac .owneruuid |

ORACLE 的语法

|  |
| --- |
| select ow.id,ow.name,ac.year,ac.month,ac.money  from t\_owners ow, t\_account ac  where ow.id(+) = ac.owneruuid |

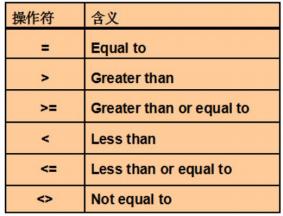
## 子查询

### where 子句中的子查询

1. 单行子查询

 只返回一条记录

 单行操作符

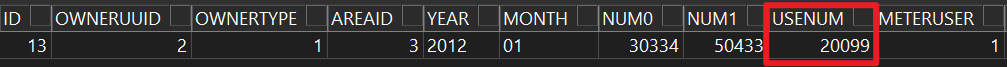


需求 ：查询 2012 年 1 月用水量大于平均值的台账记录

语句 ：

|  |
| --- |
| select \* from t\_account  where year='2012' and month='01'  and usenum>  (select avg(usenum) from t\_account where year= '2012' and month='01') |

查询结果 ：



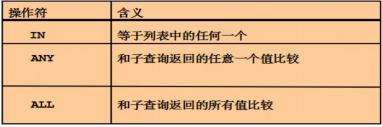
平均值为 ：



2. 多行子查询

 返回了多条记录

 多行操作符



in 运算符

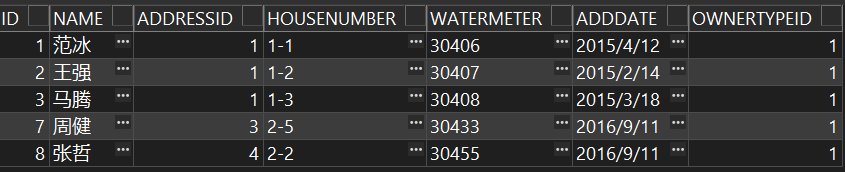
( 1 ) 需求 ：查询地址编号为 1、3、4 的业主记录

分析 ：如果我们用 or 运算符编写 ，SQL 非常繁琐 ，所以我们用 in 来进行查询

语句如下 ：

|  |
| --- |
| select \* from t\_owners  where addressid in (1,3,4) |

查询结果如下 ：



( 2 ) 需求 ：查询地址含有“花园”的业主的信息

语句 ：

|  |
| --- |
| select \* from t\_owners  where addressid in  (select id from t\_address where name like '%花园%') |

查询结果 ：

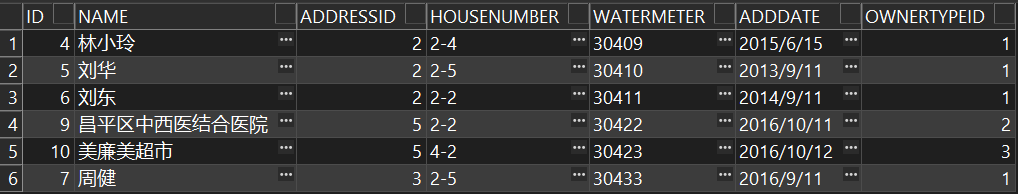


( 3 ) 需求 ：查询地址不含有“花园”的业主的信息

语句 ：

|  |
| --- |
| select \* from t\_owners  where addressid not in  (select id from t\_address where name like '%花园%') |

查询结果 ：



### from 子句中的子查询

from 子句的子查询为多行子查询

需求 ：查询显示业主编号 ，业主名称 ，业主类型名称 ，条件为业主类型为”居民” ， 使用子查询实现。

语句 ：

|  |
| --- |
| select \* from  (select o.id 业主编号, o.name 业主名称, ot.name 业主类型  from t\_owners o, t\_ownertype ot  where o.ownertypeid=ot.id)  where 业主类型='居民' |

查询结果如下:



### select 子句中的子查询

select 子句的子查询必须为单行子查询

( 1 ) 需求 ：列出业主信息 ，包括 ID ，名称 ，所属地址。

语句 ：

|  |
| --- |
| se**lect id, name,**  **(select t2.name from t\_address t2 where t2.id=t1.addressid) addressname**  **from t\_owners t1** |

查询结果如下 ：



( 2 ) 需求 ：列出业主信息 ，包括 ID ，名称 ，所属地址 ，所属区域。

语句:

|  |
| --- |
| select  t1.id,  t1.name,  (select name from t\_address where id=t1.addressid) address\_name,  (select (select t3.name from t\_area t3 where t2.areaid=t3.id ) from t\_address t2 where t2.id=t1.addressid) as area\_name  from t\_owners t1  ; |

查询结果如下 ：



## 分页查询

### 简单分页

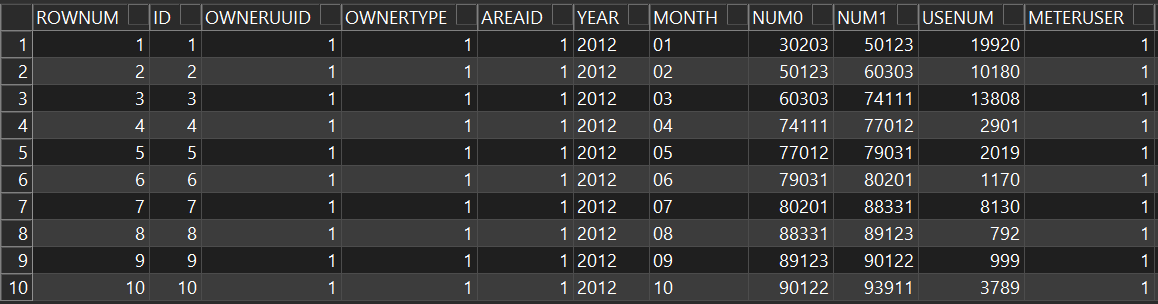
需求 ：分页查询台账表 T\_ACCOUNT ，每页 10 条记录

分析 ：我们在 ORACLE 进行分页查询 ，需要用到伪列 ROWNUM 和嵌套查询

我们首先显示前 10 条记录 ，语句如下 ：

|  |
| --- |
| select rownum, t.\* from t\_account t where rownum<=10 |

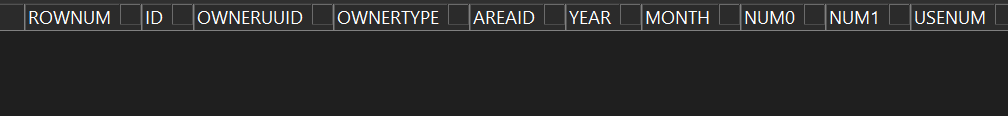
显示结果如下 ：



那么我们显示第 11 条到第 20 条的记录呢？编写语句 ：

|  |
| --- |
| select rownum,t .\* from t\_account t  where rownum>10 and rownum<=20 |

查询结果 ：



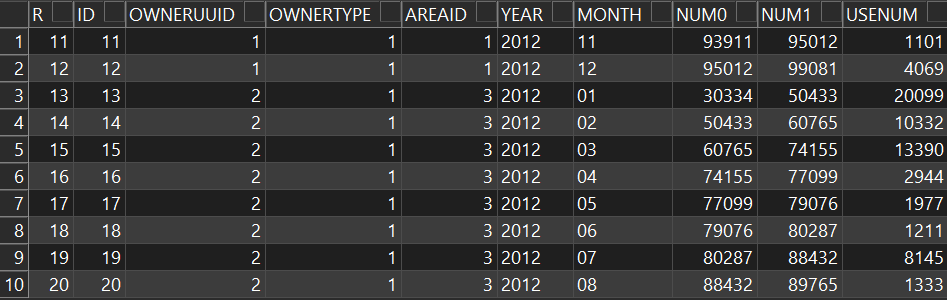
嗯？怎么没有结果？

这是因为 rownum 是在查询语句扫描每条记录时产生的 ，所以不能使用“大于”符号 ，只能使用“小于”或“小于等于”，只用“等于”也不行。

那怎么办呢？我们可以使用子查询来实现

|  |
| --- |
| select \* from  (select rownum r,t.\* from T\_ACCOUNT t where rownum<=20)  where r>10 |

查询结果如下 ：



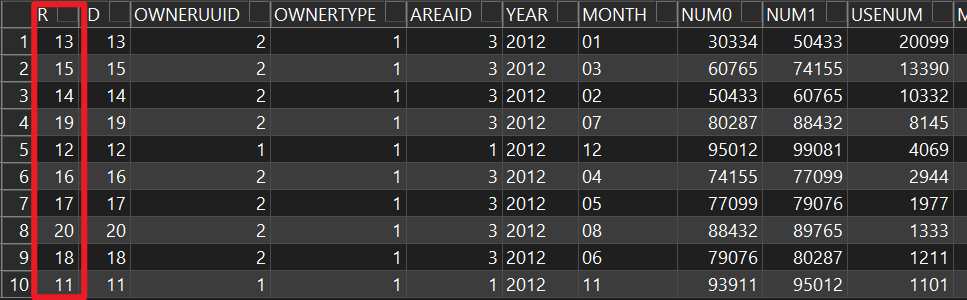
### 基于排序的分页

需求 ：分页查询台账表 T\_ACCOUNT ，每页 10 条记录 ，按使用字数降序排序。

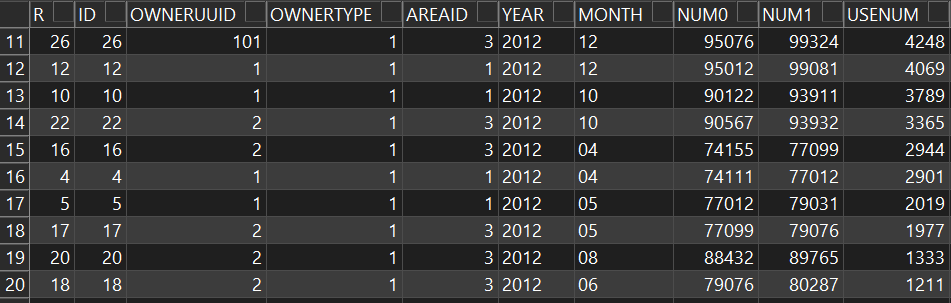
我们查询第 2 页数据 ，如果基于上边的语句添加排序 ，语句如下 ：

|  |
| --- |
| select \* from  (select rownum r,t.\* from T\_ACCOUNT t  where rownum<=20  order by usenum desc)  where r>10  ; |

查询结果如下 ：



经过验证 ，我们看到第 2 页的结果应该是下列记录

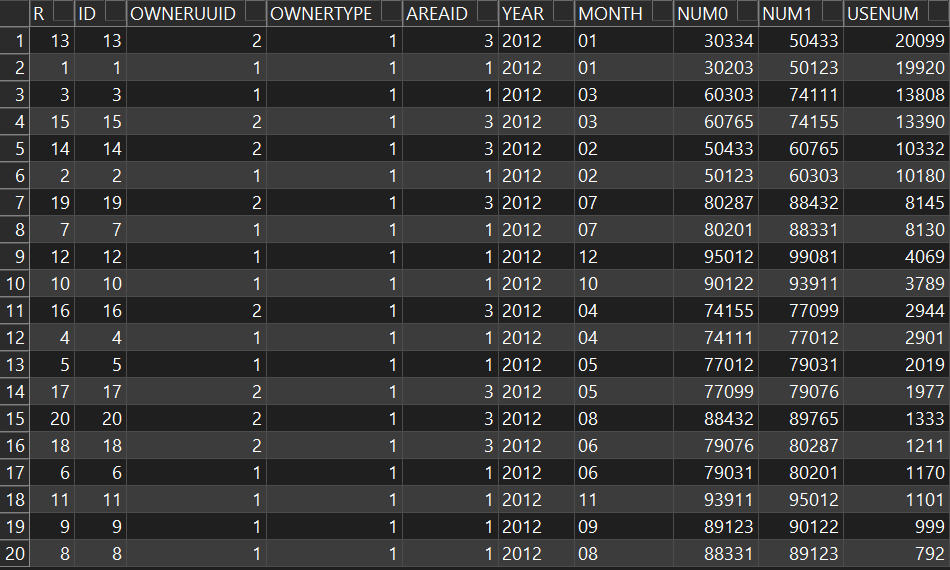


所以推断刚才的语句是错误的 ！那为什么是错误的呢 ？

我们可以先单独执行嵌套查询里面的那句话

|  |
| --- |
| select rownum r,t.\* from t\_account t  where rownum<=20 order by usenum desc |

你会看到查询结果如下 ：



你会发现排序后的 R 是乱的。这是因为 ROWNUM 伪列的产生是在表记录扫描 是产生的 ，而排序是后进行的 ，排序时 R 已经产生了 ，所以排序后 R 是乱的。

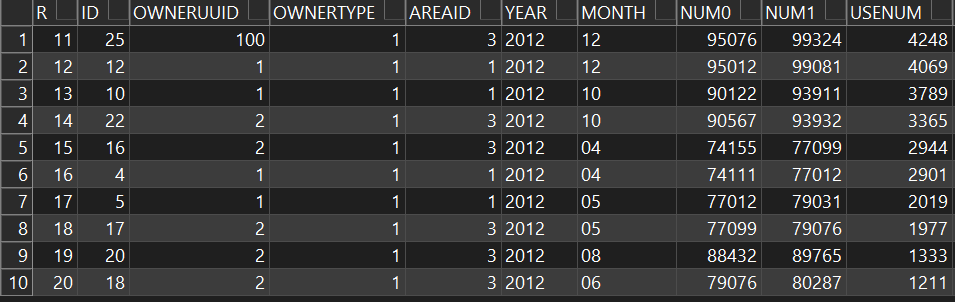
那该如何写呢 ？

很简单 ，我们只要再嵌套一层循环( 一共三层) ，让结果先排序 ，然后对排序后 的结果再产生 R ，这样就不会乱了。

语句如下 ：

|  |
| --- |
| select \* from  (select rownum r, t.\* from  (select \* from T\_ACCOUNT order by usenum desc) t  where rownum<=20)  where r>10 |

结果如下 ：



## 单行函数

### 字符函数

|  |  |
| --- | --- |
| 函 数 | 说 明 |
| ASCII | 返回对应字符的十进制值 |
| CHR | 给出十进制返回字符 |
| CONCAT | 拼接两个字符串 ，与 || 相同 |
| INITCAT | 将字符串的第一个字母变为大写 |
| INSTR | 找出某个字符串的位置 |
| INSTRB | 找出某个字符串的位置和字节数 |
| LENGTH | 以字符给出字符串的长度 |
| LENGTHB | 以字节给出字符串的长度 |
| LOWER | 将字符串转换成小写 |
| LPAD | 使用指定的字符在字符的左边填充 |
| LTRIM | 在左边裁剪掉指定的字符 |
| RPAD | 使用指定的字符在字符的右边填充 |
| RTRIM | 在右边裁剪掉指定的字符 |
| REPLACE | 执行字符串搜索和替换 |
| SUBSTR | 取字符串的子串 |
| SUBSTRB | 取字符串的子串 (以字节) |
| SOUNDEX | 返回一个同音字符串 |
| TRANSLATE | 执行字符串搜索和替换 |
| TRIM | 裁剪掉前面或后面的字符串 |
| UPPER | 将字符串变为大写 |

常用字符函数讲解 ：

( 1 ) 求字符串长度 LENGTH

语句 ：

|  |
| --- |
| select length('ABCD') from dual; |

显示结果为 ：



( 2 ) 求字符串的子串 SUBSTR

语句 ：

|  |
| --- |
| select substr('ABCD',2,2) from dual; |

显示结果为 ：



( 3 ) 字符串拼接 CONCAT

语句 ：

|  |
| --- |
| select concat('ABC', 'D') from dual; |

查询结果如下 ：



我们也可以用|| 对字符串进行拼接

|  |
| --- |
| select 'ABC' || 'D' from dual; |

查询结果同上。

### 数值函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| ABS(value) | 绝对值 |
| CEIL(value) | 大于或等于 value 的最小整数 |
| COS(value) | 余弦 |
| COSH(value) | 反余弦 |
| EXP(value) | e 的 value 次幂 |
| FLOOR(value) | 小于或等于 value 的最大整数 |
| LN(value) | value 的自然对数 |
| LOG(value) | value 的以 10 为底的对数 |
| MOD(value,divisor) | 求模 |
| POWER(value,exponent) | value 的 exponent 次幂 |
| ROUND(value,precision) | 按 precision 精度 4 舍 5 入 |
| SIGN(value) | value 为正返回 1;为负返回-1;为 0 返回 0. |
| SIN(value) | 余弦 |
| SINH(value) | 反余弦 |
| SQRT(value) | value 的平方根 |
| TAN(value) | 正切 |
| TANH(value) | 反正切 |
| TRUNC(value,按 precision) | 按照 precision 截取 value |
| VSIZE(value) | 返回 value 在 ORACLE 的存储空间大小 |

常用数值函数讲解 ：

( 1 ) 四舍五入函数 ROUND

语句 ：

|  |
| --- |
| select round(100.567) from dual |

查询结果如下 ：



语句 ：

|  |
| --- |
| select round(100.567,2) from dual; |

查询结果如下 ：



( 2 ) 截取函数 TRUNC

语句 ：

|  |
| --- |
| select trunc(100.567) from dual; |

查询结果 ：



语句 ：

|  |
| --- |
| select trunc(100.567, 2) from dual; |

查询结果 ：



( 3 ) 取模 MOD

语句 ：

|  |
| --- |
| select mod(10, 3) from dual; |

结果 ：



(三) 日期函数

|  |  |
| --- | --- |
| 函 数 | 描 述 |
| ADD\_MONTHS | 在日期 date 上增加 count 个月 |
| GREATEST(date1,date2,. . .) | 从日期列表中选出最晚的日期 |
| LAST\_DAY( date ) | 返回日期 date 所在月的最后一天 |
| LEAST( date1, date2, . . .) | 从日期列表中选出最早的日期 |
| MONTHS\_BETWEEN(date2, date1) | 给出 Date2 - date1 的月数(可以是小数) |
| NEXT\_DAY( date,’day’) | 给出日期 date 之后下一天的日期 ，这里的 day 为星期 ， 如 ：MONDAY,Tuesday 等。 |
| NEW\_TIME(date, ’this’,’other’) | 给出在 this 时区=Other 时区的日期和时间 |
| ROUND(date,’format’) | 未指定 format 时 ，如果日期中的时间在中午之前 ，则 将日期中的时间截断为 12 A.M.(午夜 ，一天的开始),否 则进到第二天。  时间截断为 12 A.M.(午夜 ，一天的开始), 否则进到第二天。 |
| TRUNC(date,’format’) | 未指定 format 时 ，将日期截为 12 A.M.( 午夜 ，一天的开始). |

我们用 sysdate 这个系统变量来获取当前日期和时间

语句如下 ：

|  |
| --- |
| select sysdate from dual; |

查询结果如下 ：



常用日期函数讲解 ：

( 1 ) 加月函数 ADD\_MONTHS：在当前日期基础上加指定的月

语句 ：

|  |
| --- |
| select add\_months(sysdate, 2) from dual |

查询结果如下 ：



( 2 )求所在月最后一天 LAST\_DAY

语句 ：

|  |
| --- |
| select last\_day(sysdate) from dual |

查询结果如下 ：



( 3 ) 日期截取 TRUNC

语句 ：

|  |
| --- |
| select trunc(sysdate) from dual |

查询结果如下 ：



语句 ：

|  |
| --- |
| select trunc(sysdate, 'yyyy') from dual |

查询结果如下 ：



语句 ：

|  |
| --- |
| select trunc(sysdate, 'mm') from dual |

查询结果如下 ：



### 转换函数

|  |  |
| --- | --- |
| 函 数 | 描 述 |
| CHARTOROWID | 将 字符转换到 rowid 类型 |
| CONVERT | 转换一个字符节到另外一个字符节 |
| HEXTORAW | 转换十六进制到 raw 类型 |
| RAWTOHEX | 转换 raw 到十六进制 |
| ROWIDTOCHAR | 转换 ROWID 到字符 |
| TO\_CHAR | 转换日期格式到字符串 |
| TO\_DATE | 按照指定的格式将字符串转换到日期型 |
| TO\_MULTIBYTE | 把单字节字符转换到多字节 |
| TO\_NUMBER | 将数字字串转换到数字 |
| TO\_SINGLE\_BYTE | 转换多字节到单字节 |

常用转换函数讲解 ：

( 1 ) 数字转字符串 TO\_CHAR

语句 ：

|  |
| --- |
| select to\_char(1024) from dual |

查询结果 ：



( 2 ) 日期转字符串 TO\_CHAR

语句 ：

|  |
| --- |
| select to\_char(sysdate, 'yyyy-mm-dd') from dual |

查询结果 ：



语句 ：

|  |
| --- |
| select to\_char(sysdate, 'yyyy-mm-dd hh:mi:ss') from dual |

查询结果 ：



语句 ：

|  |
| --- |
| select to\_char(sysdate, 'yyyy-mm-dd hh24:mi:ss') from dual |

查询结果 ：



( 3 )字符串转日期 TO\_DATE

语句 ：

|  |
| --- |
| select to\_date('2017-01-01', 'yyyy-mm-dd') from dual |

查询结果如下 ：



( 4 ) 字符串转数字 TO\_NUMBER

语句 ：

|  |
| --- |
| select to\_number('100'), '100' from dual |

查询结果如下 ：



### 其它函数

( 1 ) 空值处理函数 NVL

用法 ：

NVL (检测的值 ，如果为 null 的值) ；

语句 ：

|  |
| --- |
| select NVL(NULL, 0) from dual |

查询结果如下 ：



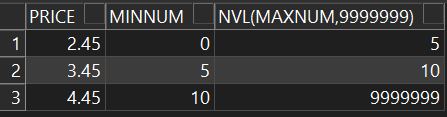
需求 ：

显示价格表中业主类型 ID 为 1 的价格记录 ，如果上限值为 NULL,则显示 9999999

语句 ：

|  |
| --- |
| select price, minnum, nvl(maxnum,9999999)  from t\_pricetable  where ownertypeid=1 |

查询结果 ：

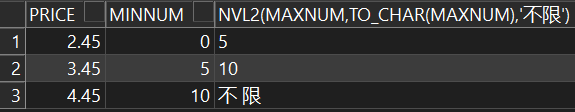


( 2 ) 空值处理函数 NVL2

用法 ：

NVL2 (检测的值，如果不为 null 的值，如果为 null 的值) ；

需求 ：显示价格表中业主类型 ID 为 1 的价格记录 ，如果上限值为 NULL,显示“不限“ .



语句 ：

|  |
| --- |
| select price, minnum, nvl2(maxnum, to\_char(maxnum), '不 限')  from t\_pricetable  where ownertypeid=1 |

( 3 ) 条件取值 decode

语法 ：

|  |
| --- |
| decode(条件,值 1,翻译值 1,值 2,翻译值 2,...值 n,翻译值 n,缺省值)  【功能】根据条件返回相应值 |

需求 ：显示下列信息 (不要关联查询业主类型表 ，直接判断 1 2 3 的值)



语句 ：

|  |
| --- |
| select name,  decode(ownertypeid, 1, '居 民', 2, '行 政 事 业 单 位', 3, '商业') as 类型  from t\_owners |

上边的语句也可以用 case when then 语句来实现

|  |
| --- |
| select name,  (case ownertypeid  when 1 then '居民'  when 2 then '行政事业单位 '  when 3 then '商业' end)  from t\_owners |

还有另外一种写法 ：

|  |
| --- |
| select name,  (case  when ownertypeid=1 then '居民'  when ownertypeid=2 then '行政事业'  when ownertypeid=3 then '商业'  end )  from t\_owners |

## 行列转换

需求 ：按月份统计 2012 年各个地区的水费 ，如下图



代码如下:

|  |
| --- |
| select  (select name from T\_AREA where id= areaid ) 区域,  sum ( case when month= '01' then money else 0 end) 一月,  sum ( case when month= '02' then money else 0 end) 二月,  sum ( case when month= '03' then money else 0 end) 三月,  sum ( case when month= '04' then money else 0 end) 四月,  sum ( case when month= '05' then money else 0 end) 五月,  sum ( case when month= '06' then money else 0 end) 六月,  sum ( case when month= '07' then money else 0 end) 七月,  sum ( case when month= '08' then money else 0 end) 八月,  sum ( case when month= '09' then money else 0 end) 九月,  sum ( case when month= '10' then money else 0 end) 十月,  sum ( case when month= '11' then money else 0 end) 十一月,  sum ( case when month= '12' then money else 0 end) 十二月  from t\_account  where year='2012'  group by areaid  ; |

需求 ：按季度统计 2012 年各个地区的水费 ，如下图



语句如下 ：

|  |
| --- |
| select  (select t2.name from t\_area t2 where id=t1.areaid) 区域,  sum(case when month in ('01', '02', '03') then money end) as 一月,  sum(case when month in ('04', '05', '06') then money end) as 二月,  sum(case when month in ('07', '08', '09') then money end) as 三月,  sum(case when month in ('10', '11', '12') then money end) as 四月  from t\_account t1  where year='2012'  group by areaid |

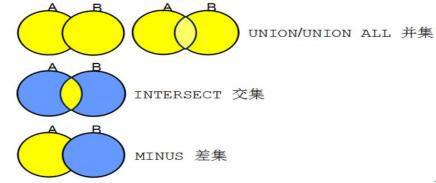
## 集合运算

### 什么是集合运算

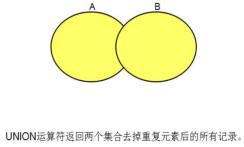
集合运算，集合运算就是将两个或者多个结果集组合成为一个结果集。

集合运算 包括 ：

* UNION ALL(并集) ，返回各个查询的所有记录 ，包括重复记录。
* UNION(并集) ，返回各个查询的所有记录 ，不包括重复记录。
* INTERSECT(交集) ，返回两个查询共有的记录。
* MINUS(差集) ，返回第一个查询检索出的记录减去第二个查询检索出的记录之 后剩余的记录。



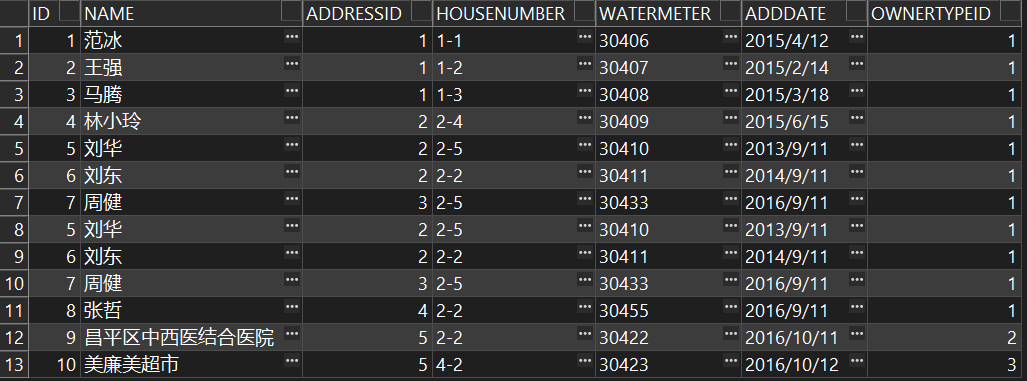
### 并集运算



UNION ALL 不去掉重复记录

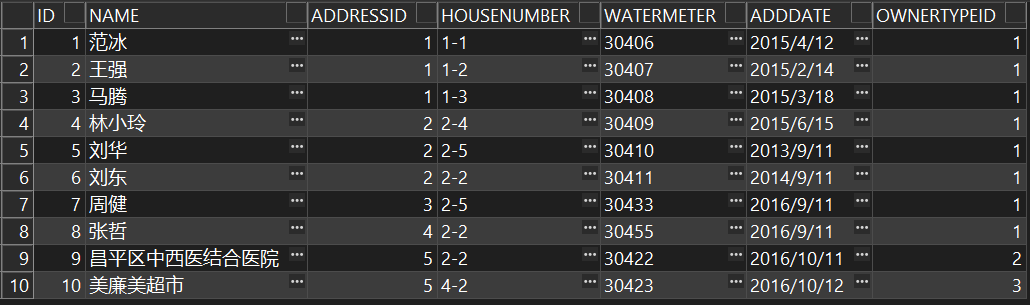
|  |
| --- |
| select \* from t\_owners where id<=7  union all  select \* from t\_owners where id>=5 |

结果如下 ：

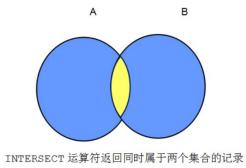


UNION 去掉重复记录

|  |
| --- |
| select \* from t\_owners where id<=7  union  select \* from t\_owners where id>=5 |

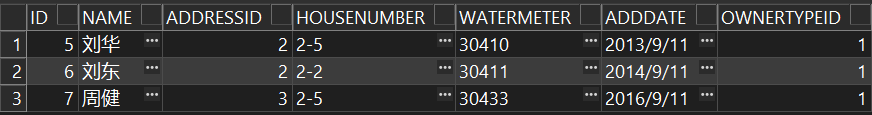


### 交集运算

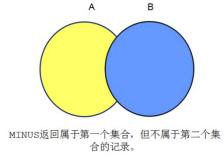


|  |
| --- |
| select \* from t\_owners where id<=7  intersect  select \* from t\_owners where id>=5 |

结果如下:

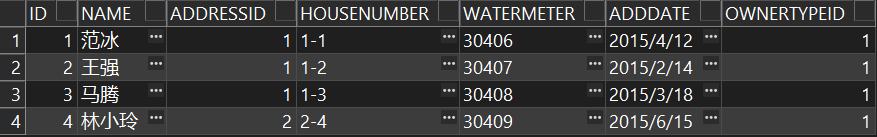


### 差集运算



|  |
| --- |
| select \* from t\_owners where id<=7  minus  select \* from t\_owners where id>=5 |

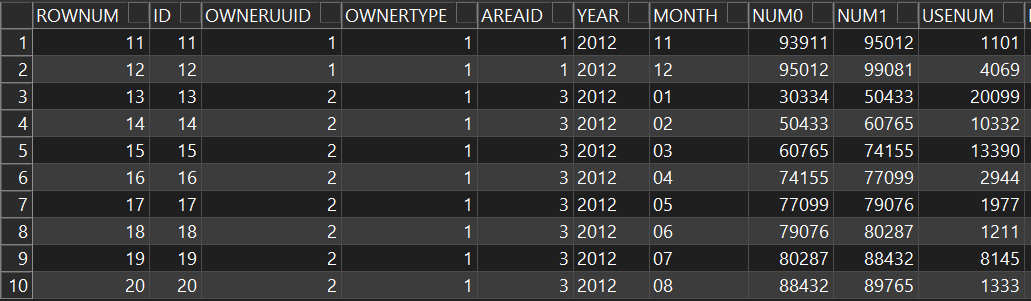
结果如下:



如果我们用 minus 运算符来实现分页 ，语句如下 ：

|  |
| --- |
| select rownum,t.\* from t\_account t where rownum<=20  minus  select rownum,t.\* from t\_account t where rownum<=10 |

结果如下:



## 分析函数(窗口函数)

## 总结

### 知识点总结

### 上机任务布置

为《自来水收费系统》开发统计模块相关的功能

1.收费日报单 (总)

统计某日的收费 ，按区域分组汇总 ，效果如下 ：



2.收费日报单 (收费员)

统计某收费员某日的收费 ，按区域分组汇总 ，效果如下 ：



3.收费月报表 (总)

统计某年某月的收费记录 ，按区域分组汇总



4.收费月报表 (收费员)

统计某收费员某年某月的收费记录 ，按区域分组汇总

5.收费年报表 (分区域统计)

统计某年收费情况 ，按区域分组汇总 ，效果如下 ：



6.收费年报表 (分月份统计)

统计某年收费情况 ，按月份分组汇总 ，效果如下



7.收费年报表 (分月份统计)

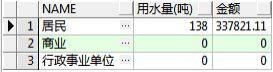
统计某年收费情况 ，按月份分组汇总 ，效果如下



8.统计用水量 ，收费金额 (分类型统计)

根据业主类型分别统计每种居民的用水量 (整数 ，四舍五入) 及收费金额 ，如

果该类型在台账表中无数据也需要列出值为 0 的记录 ， 效果如下 ：



分析 ：这里所用到的知识点包括左外连接、sum()、分组 groupby 、round() 和 nvl()

9.统计每个区域的业主户数 ，并列出合计



10.统计每个区域的业主户数 ，如果该区域没有业主户数也要列出 0

如图:

