**分析函数2(Rank, Dense\_rank, row\_number)**

**目录**

**===============================================**

**1.使用rownum为记录排名**

**2.使用分析函数来为记录排名**

**3.使用分析函数为记录进行分组排名**

**一、使用rownum为记录排名：**

在前面一篇《Oracle开发专题之：分析函数》，我们认识了分析函数的基本应用，现在我们再来考虑下面几个问题：

**①对所有客户按订单总额进行排名**

**②按区域和客户订单总额进行排名**

**③找出订单总额排名前13位的客户**

**④找出订单总额最高、最低的客户**

**⑤找出订单总额排名前25%的客户**

按照前面第一篇文章的思路，我们只能做到对各个分组的数据进行统计，如果需要排名的话那么只需要简单地加上rownum不就行了吗？事实情况是否如此想象般简单，我们来实践一下。

【1】测试环境：

SQL> desc user\_order;

 Name             Null?     Type

 ---------------------------- -------- ----------------------------

 REGION\_ID           **NUMBER**(**2**)

 CUSTOMER\_ID                   **NUMBER**(**2**)

 CUSTOMER\_SALES                  **NUMBER**

【2】测试数据：

SQL> select \* from user\_order order by customer\_sales;

 REGION\_ID CUSTOMER\_ID CUSTOMER\_SALES

----------  -----------  --------------

**5**           **1**              **151162**

**10**          **29**             **903383**

**6**           **7**              **971585**

**10**          **28**            **986964**

**9**          **21**           **1020541**

**9**          **22**           **1036146**

**8**          **16**           **1068467**

**6**           **8**            **1141638**

**5**           **3**            **1161286**

**5**           **5**            **1169926**

**8**          **19**           **1174421**

**7**          **12**           **1182275**

**7**          **11**           **1190421**

**6**          **10**           **1196748**

**6**           **9**            **1208959**

**10**          **30**          **1216858**

**5             2                1224992**

**9             24              1224992**

**9             23              1224992**

**8**          **18**           **1253840**

**7**          **15**           **1255591**

**7**          **13**           **1310434**

**10**          **27**          **1322747**

**8**          **20**           **1413722**

**6**           **6**            **1788836**

**10**          **26**          **1808949**

**5**           **4**            **1878275**

**7**          **14**           **1929774**

**8**          **17**           **1944281**

**9**          **25**           **2232703**

**30** rows selected.

注意这里有3条记录的订单总额是一样的。假如我们现在需要筛选排名前12位的客户，如果使用rownum会有什么样的后果呢？

SQL> select rownum, t.\*

**2**    from (select \*

**3**            from user\_order

**4**           order by customer\_sales desc) t

**5**   where rownum <= **12**

**6**   order by customer\_sales desc;

    ROWNUM  REGION\_ID CUSTOMER\_ID CUSTOMER\_SALES

----------  ----------  -----------  --------------

**1**          **9**                 **25**        **2232703**

**2**          **8**                 **17**        **1944281**

**3**          **7**                 **14**        **1929774**

**4**          **5**                   **4**        **1878275**

**5**         **10**                **26**        **1808949**

**6**          **6**                   **6**        **1788836**

**7**          **8**                 **20**        **1413722**

**8**         **10**                **27**        **1322747**

**9**          **7**                **13**        **1310434**

**10**          **7**               **15**        **1255591**

**11**          **8**               **18**        **1253840**

**12             5                     2          1224992**

**12** rows selected.

很明显假如只是简单地按rownum进行排序的话，我们漏掉了另外两条记录(参考上面的结果)。

**二、使用分析函数来为记录排名：**

针对上面的情况，Oracle从8i开始就提供了3个分析函数：rand，dense\_rank，row\_number来解决诸如此类的问题，下面我们来看看这3个分析函数的作用以及彼此之间的区别：

Rank，Dense\_rank，Row\_number函数为每条记录产生一个从1开始至N的自然数，N的值可能小于等于记录的总数。这3个函数的唯一区别在于当碰到相同数据时的排名策略。

**①ROW\_NUMBER：**

Row\_number函数返回一个唯一的值，当碰到相同数据时，排名按照记录集中记录的顺序依次递增。

**②DENSE\_RANK：**

Dense\_rank函数返回一个唯一的值，除非当碰到相同数据时，此时所有相同数据的排名都是一样的。

**③RANK：**

Rank函数返回一个唯一的值，除非遇到相同的数据时，此时所有相同数据的排名是一样的，同时会在最后一条相同记录和下一条不同记录的排名之间空出排名。

这样的介绍有点难懂，我们还是通过实例来说明吧，下面的例子演示了3个不同函数在遇到相同数据时不同排名策略：

SQL> select region\_id, customer\_id, sum(customer\_sales) total,

**2**         rank() over(order by sum(customer\_sales) desc) rank,

**3**         dense\_rank() over(order by sum(customer\_sales) desc) dense\_rank,

**4**         row\_number() over(order by sum(customer\_sales) desc) row\_number

**5**    from user\_order

**6**   group by region\_id, customer\_id;

 REGION\_ID CUSTOMER\_ID      TOTAL       RANK DENSE\_RANK ROW\_NUMBER

---------- ----------- ---------- ---------- ---------- ----------

**8**          **18**                **1253840**         **11**         **11**         **11**

**5**           **2**                 **1224992**         **12**         **12**         **12**

**9**          **23**                **1224992**         **12**         **12**         **13**

**9**          **24**                **1224992**         **12**         **12**         **14**

**10**          **30**               **1216858**         **15           13            15**

**30** rows selected.

请注意上面的绿色高亮部分，这里生动的演示了3种不同的排名策略：

①对于第一条相同的记录，3种函数的排名都是一样的：12

②当出现第二条相同的记录时，Rank和Dense\_rank依然给出同样的排名12；而row\_number则顺延递增为13，依次类推至第三条相同的记录

③当排名进行到下一条不同的记录时，可以看到Rank函数在12和15之间空出了13,14的排名，因为这2个排名实际上已经被第二、三条相同的记录占了。而Dense\_rank则顺序递增。row\_number函数也是顺序递增

比较上面3种不同的策略，我们在选择的时候就要根据客户的需求来定夺了：

**①假如客户就只需要指定数目的记录，那么采用row\_number是最简单的，但有漏掉的记录的危险**

**②假如客户需要所有达到排名水平的记录，那么采用rank或dense\_rank是不错的选择。至于选择哪一种则看客户的需要，选择dense\_rank或得到最大的记录**

**三、使用分析函数为记录进行分组排名：**

上面的排名是按订单总额来进行排列的，现在跟进一步：假如是为各个地区的订单总额进行排名呢？这意味着又多了一次分组操作：对记录按地区分组然后进行排名。幸亏Oracle也提供了这样的支持，我们所要做的仅仅是在over函数中order by的前面增加一个分组子句：partition by region\_id。

SQL> select region\_id, customer\_id,

               sum(customer\_sales) total,

**2**         rank() over(partition by region\_id

                        order by sum(customer\_sales) desc) rank,

**3**         dense\_rank() over(partition by region\_id

                        order by sum(customer\_sales) desc) dense\_rank,

**4**         row\_number() over(partition by region\_id

                        order by sum(customer\_sales) desc) row\_number

**5**    from user\_order

**6**   group by region\_id, customer\_id;

 REGION\_ID CUSTOMER\_ID      TOTAL       RANK DENSE\_RANK ROW\_NUMBER

---------- ----------- ---------- ---------- ---------- ----------

**5**           **4**                **1878275**          **1**          **1**          **1**

**5**           **2**                **1224992**          **2**          **2**          **2**

**5**           **5**                **1169926**          **3**          **3**          **3**

**6**           **6**                **1788836**          **1**          **1**          **1**

**6**           **9**                **1208959**          **2**          **2**          **2**

**6**          **10**               **1196748**          **3**          **3**          **3**

**30** rows selected.

现在我们看到的排名将是基于各个地区的，而非所有区域的了！Partition by 子句在排列函数中的作用是将一个结果集划分成几个部分，这样排列函数就能够应用于这各个子集。