**分析函数(OVER)**

**目录：**

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

**1.Oracle分析函数简介**

**2. Oracle分析函数简单实例**

**3.分析函数OVER解析**

**一、Oracle分析函数简介：**

在日常的生产环境中，我们接触得比较多的是OLTP系统(即Online Transaction Process)，这些系统的特点是具备实时要求，或者至少说对响应的时间多长有一定的要求；其次这些系统的业务逻辑一般比较复杂，可能需要经过多次的运算。比如我们经常接触到的电子商城。

在这些系统之外，还有一种称之为OLAP的系统(即Online Aanalyse Process)，这些系统一般用于系统决策使用。通常和数据仓库、数据分析、数据挖掘等概念联系在一起。这些系统的特点是数据量大，对实时响应的要求不高或者根本不关注这方面的要求，以查询、统计操作为主。

我们来看看下面的几个典型例子：

①查找上一年度各个销售区域排名前10的员工

②按区域查找上一年度订单总额占区域订单总额20%以上的客户

③查找上一年度销售最差的部门所在的区域

④查找上一年度销售最好和最差的产品

我们看看上面的几个例子就可以感觉到这几个查询和我们日常遇到的查询有些不同，具体有：

**①需要对同样的数据进行不同级别的聚合操作**

**②需要在表内将多条数据和同一条数据进行多次的比较**

**③需要在排序完的结果集上进行额外的过滤操作**

**分析函数语法:**

FUNCTION\_NAME(<argument>,<argument>...)

OVER (<Partition-Clause><Order-by-Clause><Windowing Clause>)

例:

sum(sal) over (partition by deptno order by ename) new\_alias

sum就是函数名

(sal)是分析函数的参数,每个函数有0~3个参数,参数可以是表达式,例如:sum(sal+comm)

over 是一个关键字,用于标识分析函数,否则查询分析器不能区别sum()聚集函数和sum()分析函数

partition by deptno 是可选的分区子句,如果不存在任何分区子句,则全部的结果集可看作一个单一的大区

order by ename 是可选的order by 子句,有些函数需要它,有些则不需要.依靠已排序数据的那些函数,如:用于访问结果集中前一行和后一行的LAG和LEAD,必须使用,其它函数,如AVG,则不需要.在使用了任何排序的开窗函数时,该子句是强制性的,它指定了在计算分析函数时一组内的数据是如何排序的.

**1)FUNCTION子句**

ORACLE提供了26个分析函数,按功能分5类

**分析函数分类**

等级(ranking)函数:用于寻找前N种查询

开窗(windowing)函数:用于计算不同的累计,如SUM,COUNT,AVG,MIN,MAX等,作用于数据的一个窗口上

例:

sum(t.sal) over (order by t.deptno,t.ename) running\_total,

sum(t.sal) over (partition by t.deptno order by t.ename) department\_total

制表(reporting)函数:与开窗函数同名,作用于一个分区或一组上的所有列

例:

sum(t.sal) over () running\_total2,

sum(t.sal) over (partition by t.deptno ) department\_total2

**制表函数与开窗函数的关键不同之处在于OVER语句上缺少一个ORDER BY子句!**

LAG,LEAD函数:这类函数允许在结果集中向前或向后检索值,为了避免数据的自连接,它们是非常用用的.

VAR\_POP,VAR\_SAMP,STDEV\_POPE及线性的衰减函数:计算任何未排序分区的统计值

**2)PARTITION子句**

按照表达式分区(就是分组),如果省略了分区子句,则全部的结果集被看作是一个单一的组

**3)ORDER BY子句**

分析函数中ORDER BY的存在将添加一个默认的开窗子句,这意味着计算中所使用的行的集合是当前分区中当前行和前面所有行,没有ORDER BY时,默认的窗口是全部的分区 在Order by 子句后可以添加**nulls last**,如:order by comm desc nulls last 表示排序时忽略comm列为空的行.

**4)WINDOWING子句**

用于定义分析函数将在其上操作的行的集合

Windowing子句给出了一个定义变化或固定的数据窗口的方法,分析函数将对这些数据进行操作

默认的窗口是一个固定的窗口,仅仅在一组的第一行开始,一直继续到当前行,要使用窗口,必须使用ORDER BY子句

根据2个标准可以建立窗口:数据值的范围(RANGES)或与当前行的行偏移量.

**5)Rang窗口**

Range 5 preceding:将产生一个滑动窗口,他在组中拥有当前行以前5行的集合

ANGE窗口仅对NUMBERS和DATES起作用,因为不可能从VARCHAR2中增加或减去N个单元

另外的限制是ORDER BY中只能有一列,因而范围实际上是一维的,不能在N维空间中

例:

avg(t.sal) over(order by t.hiredate asc range 100 preceding) 统计前100天平均工资

**6)Row窗口**

利用ROW分区,就没有RANGE分区那样的限制了,数据可以是任何类型,且ORDER BY 可以包括很多列

**7)Specifying窗口**

UNBOUNDED PRECEDING:这个窗口从当前分区的每一行开始,并结束于正在处理的当前行

CURRENT ROW:该窗口从当前行开始(并结束)

Numeric Expression PRECEDING:对该窗口从当前行之前的数字表达式(Numeric Expression)的行开始,对RANGE来说,从从行序值小于数字表达式的当前行的值开始.

Numeric Expression FOLLOWING:该窗口在当前行Numeric Expression行之后的行终止(或开始),且从行序值大于当前行Numeric Expression行的范围开始(或终止)

range between 100 preceding and 100 following:当前行100前,当前后100后

**注意:分析函数允许你对一个数据集进排序和筛选,这是SQL从来不能实现的.除了最后的Order by子句之外,分析函数是在查询中执行的最后的操作集,这样的话,就不能直接在谓词中使用分析函数,即不能在上面使用where或having子句!!!**

**二、Oracle分析函数简单实例：**

下面我们通过一个实际的例子：按区域查找上一年度订单总额占区域订单总额20%以上的客户，来看看分析函数的应用。

【1】测试环境：

SQL> desc orders\_tmp;

 Name                Null?     Type

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

 CUST\_NBR                    NOT NULL **NUMBER**(**5**) 客户号

 REGION\_ID                   NOT NULL **NUMBER**(**5**) 区域

 SALESPERSON\_ID       NOT NULL **NUMBER**(**5**) 销售员

 YEAR                      NOT NULL **NUMBER**(**4**) 年

 MONTH                     NOT NULL **NUMBER**(**2**) 月

 TOT\_ORDERS               NOT NULL **NUMBER**(**7**) 所有订单

 TOT\_SALES                  NOT NULL **NUMBER**(**11**,**2**) 所有销售额

【2】测试数据：

SQL> select \* from orders\_tmp;

CUST\_NBR  REGION\_ID SALESPERSON\_ID       YEAR      MONTH TOT\_ORDERS  TOT\_SALES

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

**11**          **7**             **11**                       **2001**          **7**          **2**      **12204**

**4**          **5**              **4**                         **2001**         **10**         **2**      **37802**

**7**          **6**              **7**                         **2001**          **2**          **3**       **3750**

**10**          **6**              **8**                        **2001**          **1**          **2**      **21691**

**10**          **6**              **7**                        **2001**          **2**          **3**      **42624**

**15**          **7**             **12**                       **2000**          **5**          **6**         **24**

**12**          **7**              **9**                        **2000**          **6**          **2**      **50658**

**1**          **5**              **2**                         **2000**          **3**          **2**      **44494**

**1**          **5**              **1**                         **2000**          **9**          **2**      **74864**

**2**          **5**              **4**                         **2000**          **3**          **2**      **35060**

**2**          **5**              **4**                         **2000**          **4**          **4**       **6454**

**2**          **5**              **1**                         **2000**         **10**          **4**      **35580**

**4**          **5**              **4**                         **2000**         **12**          **2**      **39190**

**13** rows selected.

【3】测试语句：

SQL> select o.cust\_nbr customer,

**2**         o.region\_id region,

**3**         sum(o.tot\_sales) cust\_sales,

**4**         sum(sum(o.tot\_sales)) over(partition by o.region\_id) region\_sales

**5**    from orders\_tmp o

**6**   where o.year = **2001**

**7**   group by o.region\_id, o.cust\_nbr;

  CUSTOMER     REGION CUST\_SALES REGION\_SALES

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

**4**              **5**      **37802**        **37802**

**7**              **6**       **3750**         **68065**

**10**             **6**      **64315**        **68065**

**11**             **7**      **12204**        **12204**

**三、分析函数OVER解析：**

请注意上面的绿色高亮部分，group by的意图很明显：将数据按区域ID，客户进行分组，那么Over这一部分有什么用呢？假如我们只需要统计每个区域每个客户的订单总额，那么我们只需要group by o.region\_id,o.cust\_nbr就够了。但我们还想在每一行显示该客户所在区域的订单总额，这一点和前面的不同：需要在前面分组的基础上按区域累加。很显然group by和sum是无法做到这一点的(因为聚集操作的级别不一样，前者是对一个客户，后者是对一批客户)。

**这就是over函数的作用了！它的作用是告诉SQL引擎：按区域对数据进行分区，然后累积每个区域每个客户的订单总额(sum(sum(o.tot\_sales)))。**

现在我们已经知道2001年度每个客户及其对应区域的订单总额，那么下面就是筛选那些个人订单总额占到区域订单总额20%以上的大客户了

SQL> select \*

**2**    from (select o.cust\_nbr customer,

**3**                 o.region\_id region,

**4**                 sum(o.tot\_sales) cust\_sales,

**5**                 sum(sum(o.tot\_sales)) over(partition by o.region\_id) region\_sales

**6**            from orders\_tmp o

**7**           where o.year = **2001**

**8**           group by o.region\_id, o.cust\_nbr) all\_sales

**9**   where all\_sales.cust\_sales > all\_sales.region\_sales \* **0.2**;

  CUSTOMER     REGION CUST\_SALES REGION\_SALES

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

**4**          **5**      **37802**        **37802**

**10**          **6**      **64315**        **68065**

**11**          **7**      **12204**        **12204**

SQL>

现在我们已经知道这些大客户是谁了！哦，不过这还不够，如果我们想要知道每个大客户所占的订单比例呢？看看下面的SQL语句，只需要一个简单的Round函数就搞定了。

SQL> select all\_sales.\*,

**2**         **100 \* round(cust\_sales / region\_sales, 2) || '%' Percent**

**3**    from (select o.cust\_nbr customer,

**4**                 o.region\_id region,

**5**                 sum(o.tot\_sales) cust\_sales,

**6**                 sum(sum(o.tot\_sales)) over(partition by o.region\_id) region\_sales

**7**            from orders\_tmp o

**8**           where o.year = **2001**

**9**           group by o.region\_id, o.cust\_nbr) all\_sales

**10**   where all\_sales.cust\_sales > all\_sales.region\_sales \* **0.2**;

CUSTOMER     REGION CUST\_SALES REGION\_SALES PERCENT

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

**4**            **5**                  **37802**        **37802**    **100**%

**10**           **6**                  **64315**        **68065**      **94**%

**11**           **7**                  **12204**        **12204**    **100**%

SQL>

**总结：**

**①Over函数指明在那些字段上做分析，其内跟Partition by表示对数据进行分组。注意Partition by可以有多个字段。**

**②Over函数可以和其它聚集函数、分析函数搭配，起到不同的作用。例如这里的SUM，还有诸如Rank，Dense\_rank**