**级联查询(Hierarchical Queries) 进阶应用：伪列Level**

**目录：**

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

**1.使用伪列Level显示表中节点的层次关系**

**2.统计表中节点的层数**

**3.统计表中各个层次的节点数量**

**4.查找表中各个层次的节点信息**

**5.在Start with中使用子查询**

**6.判断节点和节点之间是否具有层次关系**

**7.删除级联表中的子树**

**一、使用伪列Level显示表中节点的层次关系：**

Oracle对级联查询的支持不仅在于提供了像Start with...Connect by这样的子句供我们很方便地执行查询，而且还提供了一个伪列(Pseudocolumn): Level。这个伪列的作用是在递归查询的结果中用来表示节点在整个结构中所处的层次。下面我们来看看实际的例子：

还是上次那个employee表，现在我们要在上次的需求上面增加点小玩意：输出每个节点的层次值，看如下SQL：

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select level, id, emp\_name, manager\_id from employee

start with id = **2** connect by prior id = manager\_id order by id;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

     LEVEL         ID EMP\_NAME            MANAGER\_ID

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

**1**          **2** mark                         **1**

**2**          **4** tom                           **2**

**2**          **5** paul                          **2**

**3**          **7** ben                           **4**

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL>

我们可以看到在LEVEL列，输出了1,2,2,3的值，这就是Oracle为我们提供的一个伪列。此伪列只能用在start with...connect by子句中，下面我们来看另一种方式是否可行：

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select level, p.\* from (select \* from employee start with id = **2** connect by prior id = manager\_id order by id) p;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

     LEVEL         ID EMP\_NAME             MANAGER\_ID

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

**0**          **2** mark                          **1**

**0**          **4** tom                           **2**

**0**          **5** paul                          **2**

**0**          **7** ben                           **4**

SQL>

可以看到Level列的值全部变成了0，可见在这里Oracle并不认为虚表P里面的数据是“层次关系”，因而对于Level都返回0

**二、统计表中节点的层数：**

假设现在我们想看一下当前employee表中员工总共分为几个级别，我们应该如何做呢？请看下面的SQL

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select \* from employee;

        ID EMP\_NAME             MANAGER\_ID

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

**1** king

**2** mark                         **1**

**3** bob                           **1**

**4** tom                           **2**

**5** paul                          **2**

**6** jack                          **3**

**7** ben                           **4**

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

**7** rows selected.

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select count(level) from employee start with manager\_id is null connect by prior id = manager\_id;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

COUNT(LEVEL)

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

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

**7**

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select count(distinct level) from employee start with manager\_id is null connect by prior id = manager\_id;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

COUNT(DISTINCTLEVEL)

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

**4**

从这里我们可以看到，在统计的时候一定要使用**distinct**关键字，否则得到的错误的结果。

**三、统计表中各个层次的节点数量：**

假设我们想知道employee表中每个级别的员工数量，我们应该如何做呢--对了，使用Level和group by子句了

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select level, count(level) from employee start with manager\_id is null connect by prior id = manager\_id group by level;

LEVEL COUNT(LEVEL)

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

---------- ------------

**1**            **1**

**2**            **2**

**3**            **3**

**4**            **1**

**四、查找表中各个层次的节点信息：**

上面的例子很简单，我们看到Level可以用在group by子句中，现在我们更进一步，查看指定层次的员工信息，比如说我现在打算查看Level=2的所有员工的记录，应该如何做呢？很自然地我们想到了第一个SQL语句：

SQL> select level, id, emp\_name, manager\_id from employee where level >= **2**;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

no rows selected

很奇怪吧，这这里level关键字就不起作用了，这是因为level伪列只能在和start with...connect by子句结合时才能发挥作用，就想上面的统计各层节点数量一样，于是我们又立马想到了第二个SQL语句：

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

select \*

  from (select level, id, emp\_name, manager\_id

          from employee

         start with manager\_id is null

        connect by prior id = manager\_id

         order by id) p

 where p.level = **2**

看起来这个句子没有什么问题吧，实际执行的效果如何呢？我们在SQL\*PLUS下执行，结果却是报错：

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

ERROR at line **1**:

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

ORA-**01747**: invalid user.table.column, table.column, or column specification

很郁闷！为什么会报p.level不可识别呢？这是因为level是Oracle的伪列，并不属于任何一个表，我们必须使用别名把这个伪列“伪装”成一个实际的列，现在我们看第三个语句，注意语句高亮处。

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select \*

**2**    from (select level emp\_level , id, emp\_name, manager\_id

**3**            from employee

**4**           start with manager\_id is null

**5**          connect by prior id = manager\_id

**6**           order by id) p

**7**   where p.emp\_level = **2**;

 EMP\_LEVEL       ID EMP\_NAME         MANAGER\_ID

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

**2**           **2** mark                          **1**

**2**           **3** bob                            **1**

这次终于搞定了！不过实际上我们有更简单的解决方法，请看第四个SQL语句：

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select level, id, emp\_name, manager\_id

**2**    from employee

**3**   where level = **2**

**4**   start with manager\_id is null

**5**  connect by prior id = manager\_id

**6**   order by id;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

     LEVEL         ID EMP\_NAME             MANAGER\_ID

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

**2**          **2** mark                          **1**

**2**          **3** bob                            **1**

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

上面我们是查看某个层次的所有节点信息，现在我们打算看看所有层次的节点信息，而且要求用一种直观的信息显示出来。下面的例子演示了如何使用空格缩进的方式来直观显示节点之间的层次关系：



SQL> select level, id, lpad('  ', **2** \* (level - **1**)) || emp\_name name, manager\_id

**2**    from employee

**3**   start with manager\_id is null

**4**  connect by prior id = manager\_id;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

     LEVEL         ID NAME                 MANAGER\_ID

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

**1**          **1** king

**2**          **2**   mark                        **1**

**3**          **4**     tom                       **2**

**4**          **7**       ben                     **4**

**3**          **5**     paul                      **2**

**2**          **3**   bob                         **1**

**3**          **6**     jack                      **3**

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

**7** rows selected.

请注意这里的lpad函数的作用，正是它利用了层次和空格进行缩进，让我们可以很直观地从NAME字段对齐方式就知道各个节点的层次关系。如果我们需要过滤其中的某些节点，只需要将where条件加在start with**前面**就可以了(注意必须是前面，否则会报语法错误)。

**五、在Start with中使用子查询：**

在前面我们看到的例子中，start with的值都是一个固定的内容，但有些时候查询的起始点并不容易确定，比如：查询工号最小的员工节点及其子节点，这个时候工号最小很明显是一个查询的条件，需要我们先通过执行一个查询得到确定的值，再作为查询的起点。请看例子：

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> select level, id, lpad('  ', **2** \* (level - **1**)) || emp\_name name, manager\_id

**2**    from employee

**3**   start with id = (select min(id) from employee)

**4**  connect by prior id = manager\_id;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

     LEVEL         ID NAME                 MANAGER\_ID

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

**1**          **1** king

**2**          **2**   mark                       **1**

**3**          **4**     tom                       **2**

**4**          **7**       ben                     **4**

**3**          **5**     paul                      **2**

**2**          **3**   bob                         **1**

**3**          **6**     jack                      **3**

**7** rows selected.

**六、判断节点和节点之间是否具有层次关系：**

在日常工作中除了查询节点的信息之外，另一个常见的应用就是判断某个节点和另外一个/些节点之间是否具有层次关系。例如我想知道员工mark是不是员工jack的领导(直接或间接的都可以)，我应该怎么做呢？

考虑到start with...connect by会返回一棵节点树，假如节点数上没有jack节点，那么说明mark并不是jack的直接或间接领导，如果找到那说明mark是jack的父节点。方法简单



SQL> select level,

**2**         id,

**3**         lpad('  ', **2** \* (level - **1**)) || emp\_name employee\_name,

**4**         manager\_id

**5**    from employee

**6**   where emp\_name = 'jack'

**7**   start with emp\_name = 'mark'

**8**  connect by prior id = manager\_id;

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

no rows selected

**七、删除级联表中的子树：**

假设现在employee表中的mark及其下属员工离职，那么我们为了维护数据的完整性，必须将mark及其下属员工的节点都删除，有了start with...connect by和level我们就可以轻松地做到这一点了。

【1】按名称删除节点树：

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

SQL> delete from employee

**2**   where id in (select id

**3**                  from employee

**4**                 start with emp\_name = 'mark'

**5**                connect by prior id = manager\_id);

C:\EB5C6AC5\26C5FB6B-61EE-40DD-9128-26E6DDB44795.files\image001.gif

**4** rows deleted.