树结构数据在关系数据库中的表示与查询方法

马玉枫　（武汉测绘科技大学）

摘要：本文讨论了关系数据库中树结构信息的二维表表示方法以及在 *SQL* 查询中的应用。包括树结构的定义、根结点确定、遍历方向的选择以及条件剪枝和输出格式定义等方面的应用。

一、引言

关系数据库是当前应用最为广泛的数据库系统，其产品和工具从 dBASE 系列、Fox BASE 系统到目前流行的 ORACLE 产品。在数据处理、系统管理以及用户界面等各个方面，均提供了强大的功能支持和极为便利、丰富的工具环境。

关系数据库中最基本的数据结构是表，概念之间的关系用表的二维关系来表示。以往的关系数据库产品在对多层次关系的数据表示和处理方面没有提供相应支持，要处理这类信息将带了许多冗余，在查询方面也有不少局限。而 ORACLE 关系数据库，除了提供关系数据库的各种功能，包括 SQL 语言等第四代工具支持外，还在层次结构信息的表示和处理方面提供了支持。通过逻辑表达式描述概念结点间的关系，实现了在表中表示具有树结构的层次信息，从而优化和扩展了关系数据库的表处理功能。下面就对树结构信息的表示，以及在此基础上的相关查询作详细讨论，并给出一个例子来说明 ORACLE 对树结构信息的支持在维护数据库一致性方面的应用。

二、树结构信息的概念

在实际应用中，树结构信息是普遍存在的，如一个单位的机构谱系图，一个产品与它各个零部件的关系等，均可以用树结构来表示。图 1 给出了某个产品的组成结构示意图，是一个典型的树结构。

可见，在树结构中，每个结点与其它结点的从属关系很清楚。结点按它所处的位置分为父结点、子结点和叶结点，而父结点和子结点也是相对的。一个结点可是某个结点的子结点，又同时为另一个结点的父结点。树的遍历有多种算法，ORACLE 以如下遍历算法来扫描结点，一般步骤如下：

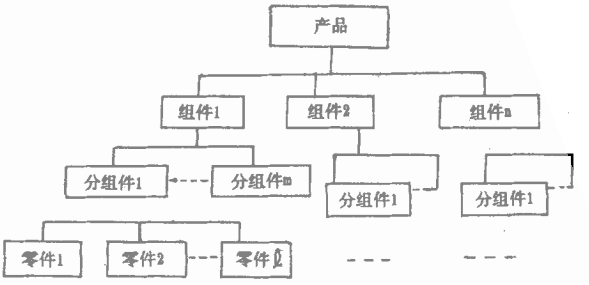

图 1

(1)从根结点开始。

(2)扫描这个结点。

(3)若该结点有尚未访问到的子结点，移到最左边未访问的子结点，转第 2 步；否则做第 4 步。

(4)如果这是根结点，结束；否则做第 5 步。

(5)返回到结点的父结点，转第 3 步。

图 1 所示的例子经上述扫描后得到如下结点序列：产品、组件 1、分组件 1、零件 1、…零件 L、分组件 2、…分组件 m、组件 2…

三、ORACLE 中树结构信息的表示

ORACLE 中基本的数据结构是表，对于树结构的信息，如何用二维表来体现出父、子结点间的连接关系呢？ORALCE 通过在表中定义记录行之间有关系来表示。

首先建立一个样本表，其中包含了某个部门的人员有关信息，如表 1 所示。用“直属关系(LID)”列来表示任一记录的父结点。例如，刘英的工作证编号(TID)为 2105，该项可以在陈波的 LID 列中找到，因此，表示陈波记录的父结点是刘英这个记录。而张平的 TID 值为 2038 可在刘英的 LID 列中找到，故张平记录为刘英记录的父结点。以此类推，通过 LID 与 TID 的关系，样本表中体现了如下树结构：

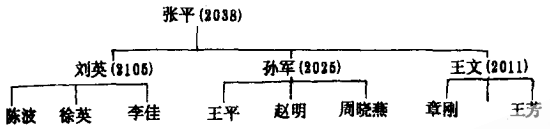

图 2

表 1

TID	姓名	性别	年龄	职位	LID	部门
2038	张平	男	45	总经理		
2056	陈波	男	29	检验员	2105	生产部
2025	孙军	男	33	经 理	2038	技术部
2177	李佳	女	30	会 计	2105	生产部
2081	章刚	男	31	销售员	2011	销售部
2011	王文	男	36	经 理	2038	销售部
:	:	:	:	:	:	:
:	:	:	:	:	:	:

对于这种树结构，OPACLE 在查询语句 SELECT 中引入 CONNECT 和 START 子句，用 CONNECT 引导一个逻辑表达式来定义父、子结点关系，以 START 子句来表示当前根结点(查询起点)。例如，

```
CONNECT BY PRIOR TID=LID
```

该子句表示：“如果结点 1 的 TID 等于结点 2 的 LID，则结点 1 为结点 2 的父结点。其中 PRIOR 表示搜索方向为从根至下，先扫描到的结点为父结点。

```
START WITH 姓名='张平'
```

子句则表示以张平结点为根开始遍历树。

我们对表 1 所示信息的查询可以用下列语句进行，如果这里的“直属关系(LID)表示直接领导”时，可列出人员及其直接领导人之间的关系(表名为 TLIST)：

例 1
```
SELECT 姓名, TID, 职位, LID
FROM TLIST
CONNECT BY PRIOR TID=LID
START WITH 姓名='张平';
```

这样返回的值与原表中顺序不同，接遍历原则，应该是张平、刘英、陈波、徐英、李佳、孙军…王芳。

除了上述基本方式外，灵活应用 ORACLE 的查询技巧，还可以完成多种多样的树结构查询。

四、利用树结构的查询

还可以通过选择起始根结点来查询部分树枝，也可以指定搜索方向从叶结点向根结点进行反向搜索。此外还可应用条件子句和格式语句来修饰输出结果，下面将分别举例说明这些查询技巧。

1.从指定根开始查询

如前所示，用 START WITH 子句标识根结点，如例 1 中，START WITH 姓名='张平' 即指定从张平记录开始搜索。事实上，ORACLE 支持从任一结点开始搜索，这实际上实现了对整个表的某一部分进行快速查询。

例 2 列出由孙军所领导的各位人员的信息。

```
SELECT 姓名, TID, LID, 职位
FROM TLIST
CONNECT BY PRIOR TID=LID
START WITH 姓名='孙军';
```

结果返回记录为：孙军、王平、赵明和周晓燕四个记录，它们是树的一部分(树枝)。

START WITH 子句还可以引导逻辑表达式，如下例即接领导--成员的形式分别列出刘英和孙军领导的小组人员，信息显示按组进行，非常清晰。

例 3
```
SELECT 姓名, TID, LID, 职位, 部门
FROM TLIST
CONNECT BY PRIOR TID=LID
START WITH 姓名='刘英' OR 姓名='孙军'
ORDER BY 部门;
```

2.选择遍历方向

一般地对树的遍历是从根开始向下进行的，但也可以从叶向上搜索。可以用 CONNECT BY 子句中的 PRIOR 算符来指定方向，把 PRIOR 置于子结点前即

可。注意，这种反向搜索不能称为遍历，因为从某个叶结点出发，只能搜索到与之相关的部分结点，而非遍历树的每个结点。

如果要查找样本表中章刚的各级领导者，可以用下列查询语句来实现：

```
例 4  SELECT  姓名，TID，LID，职位，部门
      FROM  TLIST
      CONNECT  BY
      START  WITH  姓名='章刚'
```

结果返回章刚的直接领导王文以及王文的领导张平的记录。

反向搜索可以从指定的任一结点向上进行，这为维护数据库的一致性提供了一种很好的手段。看图 1 所示的例子，如果把每个零件所包括的生产工序所用的时间值加入作为叶结点，这就是一个生产管理中有重要意义的工时定额管理的例子。要对每个零件、分组件、组件和产品逐级进行工时定额汇总，定额的制定与维护是基础。每当某个工序的定额值改变时，将逐级影响到与之相关的各级数据。要想保持数据一致性，一般的(不提供树结构及反向搜索)关系数据库只能用二种方式来进行。第一，全部进行重汇总；第二，为每个记录建立相关表，据表来修改相应的数据项。这些方法在空间上和时间上的耗费均是不可接受的，尤其对大型的产品。因此，通常只能由系统给出少量提示信息，用户用手工修改，可靠性很低。

如果用表先定义了树结构，实际上只需加入一个字段来描述“相关关系”，并利用树的反向搜索，就可以很容易地找到与任一被修改的数据项相关的各级所有数据，实现相关数据的修改。实际上，相关数据的连锁查询和维护在信息管理系统中是非常普遍存在的，树结构信息的表示与处理为这类不易维护的相关数据，提供了一个很好的解决途径。

3.使用条件子句进行剪枝

在树结构信息的查询中，可以使用 WHERE 条件子句，用逻辑表达式来控制取树的哪些结点。现通过下面的例子来说明两种不同的修剪方式。

```
例 5  SELECT  姓名，TID，LID，职位，部门
      FROM  TLIST
      WHERE  姓名!  ='王文'
      CONNECT  BY  PRIOR  TID=LID
      START  WITH  姓名='张平'
```

这个查询返回的结果是在张平领导下，除了王文以外的所有职员的记录，包括由王文领导的章刚和王芳的记录。因为 WHERE 子句是针对所有取出的记录而加上的条件选择，所以，它只能消除指定的结点，而不能排除“挂”在该结点上的各级子结点。

如果想要修剪某个树枝，即排除某个结点及其所有的子结点，可以在 CONNECT BY 子句中使用条件表达式，见例 6。

```
例 6  SELECT  姓名，TID，LID，职位，部门
      FROM  TLIST
      WHERE  姓名!  ='王文'
      CONNECT  BY  PRIOR  TID=LID
      AND  姓名!  ='王文'
      @ START  WITH  姓名='张平'  ;
```

这里返回的结果是张平领导的除了王文负责的那个部门的所有职员(加上王文在内)的其他全体职员的记录。因为这里在 CONNECT BY 子句中加入了另一个逻辑表达式：姓名! ='王文'，它指定所有取出的父结点中不包括王文结点，换言之，只有父结点不为王文的那些记录才满足条件。例 5 与例 6 的执行效果用图 3 可以很直观地说明。

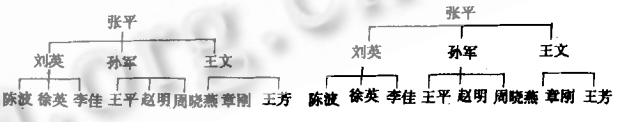

图 3

可见，WHERE 子句修剪单个结点，而 CONNECT BY 子句中的条件表达式则修剪整个树枝。

综上所述，带有多种子句的树结构查询，其执行顺序为：

(1)由 START WITH 找到根结点记录。

(2)据 CONNECT BY 子句所定义的关系构造树。

(3)按 PRIOR 所指定的方向遍历树。

(4)用 CONNECT BY 子句中的条件修剪树枝。

(5)用 WHERE 子句中的条件修剪单个结点记录。

(6)接 ORDER BY 子句对选出的记录排序。

五、树结构信息的输出格式定义

为了使信息的输出也体现出层次结构关系,可以用多种方式来定义输出格式,这里给出加级别号和缩进格式两种方法。

1.设置级别号

ORACLE 的 SQL 查询输出是以表的形式给出的,各个记录占一行顺序排列。为了能体现图 2 所示的那些层次关系,可以分别给它们加入级别号。即张平记录为第一级,刘英、孙军和王文记录为第二级,其条为第三级,以此类推。ORACLE 系统本身已提供了这个功能,通过一个为列 LEVEL,来标识树中结点离根结点的距离,使输出结果更为清晰。用户只需要查询语句中列出 LEVEL 列即可。

例 7
```
SELECT LEVEL, 姓名, TID, LID, 职位, 部门
  FROM TLIST
  CONNECT BY PRIOR TID=LID
  START WITH 姓名='张平';
```
这里返回了各个职员的信息,包括级别号。

2.定义缩进格式输出

为了使结果更直观,可以定义缩进格式,即每增加一级,其记录的起始字段缩进若干空格,形成一个齿形表,则可以利用 LPAD 函数来实现。

例 8
```
SELECT LPAD(' ',2*LEVEL),姓名,
LEVEL,部门
  FROM TLIST
  CONNECT BY PRIOR TID=LID
  START WITH 姓名='张平';
```
在此,以 LEVEL 的值乘以 2 计算每行记录前的空格数,即 LEVEL 值每大一级,姓名字段前面就多两个空格,使结果显示呈缩进形式。

姓名	LEVEL	部门
张平	1	
刘英	2	生产部
陈波	3	生产部
徐英	3	生产部
李佳	3	生产部
孙军	2	技术部
王平	3	技术部

当记录很多,层次较多时,为了保证第一列有足够的宽度供缩进用,可以用 COLUMN 命令设置第一列的输出宽度(实际的数据库中不必定义过宽,只需考虑第一列本身的需要)。

例如,把姓名字段的输出宽度定义为 25 个字符长度:SQL> COLUMN 姓名 FORMAT A25;这样,就有足够的空间用于以缩进格式显示第一列(姓名字段)。用户还可以根据需要,设计自己满意的其它格式来反映信息的层次性。

六、结语

ORACLE 产品为用户提供了极为丰富、强大的功能支持。作为关系数据库系统,通过引入表中树结构数据的表示和处理机制,很好地解决了层次信息的表示、查询和修改等问题,充分利用表对树结构数据处理的支持,为解决相关数据的[illegible]提供了很好的途径。