非定深度树与 Oracle 的等级查询

A method for processing tree－structured data of non－fixed level in oracle

张庆丰　（广州暨南大学计算机系　510632）

摘要：本文提出了一种在 Oracle 数据库中处理非定深度树型结构数据的方法。该方法利用树型数据的双亲表示法将数据存储在单个表中，充分利用 Oracle 数据库的等级查询优点，通过简单的 SQL 语句实现数据的查询、插入、删除和更新等处理。该方法能够有效存储、处理非定深度的树型结构数据。

关键词：树型结构数据　Oracle　等级查询

1　前言

作为关系型数据库的代表，Oracle RDBMS 对关系型数据有着强大的处理能力。但在不少系统中要处理树型结构的数据，或者称为等级数据。这些数据如果有固定的深度，可以对树中每一层的数据建立一个表，从而在 Oracle 中形成一个简单的关系表，可以较好的处理。如果树的深度不定，也就是说，随情况的不同，树的深度可能增加或者减少，那么采用每层建立一张表的做法，不仅浪费存储空间，而且不利数据处理。在某装备信息管理系统中，我们就遇到这样的问题。就此我们提出了一种在 oracle 9i 中存储处理该类型数据的方法。

2　非定深度的树在 Oracle 中的存储和处理

如图 1 所示，一个树有结点 n 个，其深度可变，每个结点的属性一样。不妨假设每个结点有两个属性 NodeName 和 NodeValue，根据数据结构的知识可知：如果记录每个结点的序号、属性值和父结点序号就可以完整记录整个树，并给以处理。因此可以在 Oracle 中建立一张表 tree(ID,NodeName,NodeValue,ParentID)，该表中的四个字段分别表示某一结点的结点 ID、结点名称、结点值、父结点 ID。

将所有结点的数据放入该表，就构成了单表的存储方式。该方式比每层建立一张表的多表方式要节省存储空间，这是因为 Oracle 中的每张表都要预配置一定的存储空间，在合理的情况下，单表比多表要少用空间。另一方面，对于深度不定的树，多表存储中到底需要多少个表也是问题。

对于树的处理，主要在于查询、添加、更新、删除的操作。根据数据结构的知识知道：树的结点操作多包含迭代计算，如果用编程语言来实现比较复杂，而且效率也无法保证。这里利用 Oracle 数据库的等级查询能力来实现，方法简单、效率高。下面简述单表存储的树型数据的处理语句。

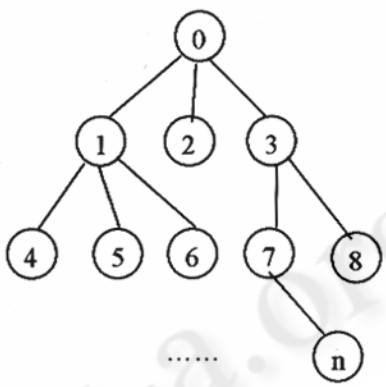

图 1　具有 n 个结点的树，其深度可变

树型数据的查询操作，主要包含兄弟结点查询、子树结点查询、父结点查询。它们的语句对应如下（其中 id1 指定要查询结点的 ID）：

```
/* 结点 id1 的所有子树结点查询 */
SELECT id, nodename, nodevalue, PRIOR id AS "
Parent ID"
FROM tree
START WITH parentid = id1
CONNECT BY parentid= PRIOR id;
/* 结点 id1 的兄弟结点查询 */
SELECT Id, nodename, nodevalue, level
FROM tree
WHERE LEVEL=( SELECT LEVEL from tree WHERE
```

```
id= id1 START WITH parentid IS NULL CONNECT BY
parentid= PRIOR id)
START WITH parentid IS NULL
CONNECT BY parentid = PRIOR id;
/* 结点 id1 的父结点查询 */
SELECT id, nodename, nodevalue
FROM tree
WHERE id=(SELECT parentid FROM tree
WHERE id=id1);
```

在以上三个语句中，使用了 Oracle 等级查询关键词。关键词 START WITH 用来指定树中查询开始的结点记录。关键词 CONNECT BY 用来指定父结点记录和子结点记录的关系，在指定中必须使用关键词 PRIOR，它表示一种运算，指定父记录和子记录之间的参照。关键词 LEVEL 是一个函数(Oracle 中称为伪列)，它返回以查询结点为根开始的层数，根为第 1 层。子树结点查询语句得出查询结点下的所有子树的结点。兄弟结点查询语句得到和查询结点在同一层的所有结点，需要注意的是其中假设树的根结点的父结点 ID 是 NULL，每一次计算 LEVEL 时是从根开始算起，其实这并不必要，如果从查询结点上面某一层的父结点开始算起会更好。父结点查询语句比较简单，直接得到查询结点对应的父结点信息。在实际处理中，也有可能需要查找从根到查询结点的路径，可以采用如下的语句：

```
/* 结点 id1 的路径查询 */
SELECT SYS_CONNECT_BY_PATH (nodename ,'/') AS
"结点路径"
FROM tree
WHERE id = id1
START WITH parentid IS NULL
CONNECT BY parentid = PRIOR id;
```

该语句将从根结点到查询结点的路径用结点名称间隔'/'来表达。

树型数据的插入，是指子树的插入，首先要确定子树中的每个结点的结点 ID 和父结点 ID，需要注意的是子树的根的父结点 ID，对应了要插入的地方。这些数据确定之后，很容易利用 insert into 语句直接插入，和一般记录的添加没有任何区别。

树型数据的更新操作，一般比较简单，对于结点属性值的更新，结点的父结点 ID 更新，和一般记录的更新操作一样，没有区别。如果更新结点的结点 ID，要稍微复杂一些，除了要更新结点的 ID，还要更新其子结点的父结点 ID。不妨假定要更新结点 ID 为 id1 的结点，其新的结点 ID 为 id2。那么如下语句可完成结点 ID 的更新。

```
/* 首先更新子结点的父结点 ID */
UPDATE tree SET parentid=id2 WHERE parentid=
id1;
/* 其次更新结点的结点 ID */
UPDATE tree SET id=id2 WHERE id=id1;
```

树型数据的删除操作，是指某个结点子树的删除。如果要删除的是结点 ID 为 id1 的子树，该操作可利用如下语句完成：

```
DELETE FROM tree START WITH parentid = id1
CONNECT BY parentid= PRIOR id;
```

3 结论

本文介绍了树型数据在 Oracle 数据库的一种存储方法，并巧妙地利用了 Oracle 提供的等级查询语句，简单、高效地实现了对这些数据的处理。该存储处理方案在某装备信息管理系统中已经得到应用，实践表明效果良好。

参考文献

1 Oracle Corporation. Oracle 9i SQL reference, 2002,10.

2 江涛、钟宏，数据结构，科学出版社，1995.

3 Steven Feuerstein, Bill Pribyl. O'Reilly: Oracle PL/SQL 程序设计(第二版)，，林琪、王宇(翻译)，中国电力出版社，2003。