

# 树型层次结构数据中遍历子树结点的方法

何 健,王家华

(西安石油大学 计算机学院,陕西 西安 710065)

**摘 要:**复杂关系数据的处理在数据库存储技术中具有重要的意义。在储层建模过程中高效的复杂数据处理方法将会大大提高建模的速度和准确性。阐述了存储过程和用户自定义函数相关概念,并以河流相随机游走储层建模相关数据为例,提出了通过使用存储过程和用户自定义函数对树型层次结构数据进行子结点递归遍历查询、存储的新方法,以便达到更高效的数据查询处理能力,最后通过 SQL 编程加以实现。实践表明,该方法具有很好的效果。

**关键词:**存储过程;数据库;SQL server

中图分类号:TP39

文献标识码:A

文章编号:1673-629X(2008)04-0095-03

## Method of Traversing Subtree Nodes in Data of Tree - Level Structure

HE Jian, WANG Jia-hua

(Computer School, Xi'an Shiyou University, Xi'an 710065, China)

**Abstract:** Complex relational data processing has the vital significance in the database storage technology. The highly effective complex relations data processing method will greatly enhance the speed and the accuracy in the reservoir modeling process. As an example of random walk reservoir modeling relevant data, through the using of server stored procedures and user-defined functions, proposed a new method that inquire and store data of subtree nodes of tree structure in order to make the more highly effective database inquiryability. Finally, through realization of the programming and relevant application, this method has the very good effect.

**Key words:** stored procedure; database; SQL server

### 0 引言

数据的存储处理是各种企业应用的关键,随着关系型数据库的发展与成熟,越来越多的海量数据得到了存储和处理。目前的关系型数据库都提供了很强大的功能,如 SQL Server 和 Oracle 都支持事务处理、触发器、存储过程和用户自定义函数等。但由于数据量的增加,其数据之间的关系复杂度也呈几何级的增加,如何进行数据筛选和处理,特别是如何更高效地对复杂树型关系数据进行遍历筛选和处理,是一个难点。以河流相随机游走储层建模相关数据为例,一个区域地质构造模型数据分为十几个小层,而每一个小层包括测井数据、井位数据、沉积相数据,其中测井数据又包括孔隙度、饱和度、渗透率等物性数据,井位数据包括油井三维坐标、井名等。这种区域构造模型、小层、物性数据之间的关系可以用一个清晰的“树形”视图表示

出来(见图1),很类似于 Windows 的资源管理器目录视图。

文中使用 SQL Server 2000 数据库服务器的存储过程及用户自定义函数两种方法进行树型结点数据遍历,从而实现效率更高、速度更快的数据处理能力。

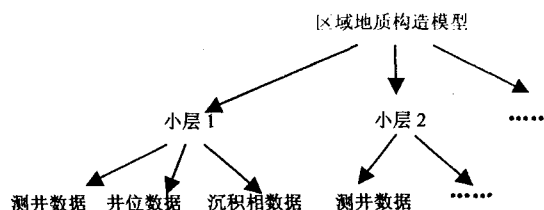


图1 地质储层数据关系图

### 1 存储过程与用户自定义函数

#### 1.1 存储过程

存储过程是 SQL 语句和可选控制流语句的预编译集合,以一个名称存储并作为数据库中的一个重要对象处理。

存储过程存储在数据库内,可由应用程序或用户直接或间接调用执行,而且允许使用声明变量、有条件执行以及其强大的编程功能<sup>[1]</sup>。存储过程包含程序

收稿日期:2007-08-14

基金项目:国家自然科学基金资助项目(50474042)

作者简介:何 健(1978-),男,硕士,主要从事储层建模及计算机图形图像方面的研究;王家华,硕士生导师,主要从事储层建模、决策分析、风险分析、石油软件方面的研究。

流、逻辑以及对数据库的查询。它们可以接受参数、输出参数、返回单个或多个结果集以及返回值。数据库中使用存储过程具有很多优点:

(1) 数据处理效率有了很大提高,特别是在 C/S 体系结构中,当多个客户端向数据库服务器提出一连串复杂的 SQL 数据处理请求,就可能导致网络运行的阻塞并引起数据库服务器的崩溃。

存储过程是在数据库服务器端执行的 SQL 代码集,客户端只需发出简单的调用请求,这就大大减少了网络带宽阻塞。并使整个数据库服务器运行性能得到显著的改善。

(2) 减少客户端 SQL 硬代码的编写,增加可重用性、移植性和可维护性。一个存储过程用于完成一个特定的任务,如数据库触发器、应用程序或其它存储过程可能需要调用该过程,从所有这些方面均可调用同一个存储过程,这样可降低软件维护的成本,减少因为客户端因需求变化而引起的代码改变。

(3) 执行速度快,减少网络流量。存储过程在创建时就经过了语法检查和性能优化,因此在执行时不必再重复这些步骤。存储过程在经过第一次调用后,就驻留在内存中,不必再经过编译和优化,所以执行速度快。在有大量批处理的 SQL 要重复执行的时候,使用存储过程可以极大地提高运行效率。

(4) 增加数据库数据的安全性。基于面向对象的方法,防止用户直接访问数据库中的表,强制用户使用存储过程执行特定的函数,并通过函数进行存储过程的管理,这比直接管理表和字段要好。

## 1.2 自定义函数

用户定义的函数(UDF)是由一个或多个 Transact-SQL 语句组成的子程序,可用于封装代码以便重新使用。它可以接受参数,处理逻辑,然后返回某些数据。SQL Server 2000 并不将用户限制在定义为 Transact-SQL 语言一部分的内置函数上,而是允许用户创建自己的用户定义函数<sup>[2]</sup>。UDF 的一个关键特征是返回值取决于 UDF 类型,调用例程可以使用这个值来继续处理它的数据。因此,如果 UDF 返回单一标量值,调用例程就可以使用标准变量或文字值的地方使用这个值。如果 UDF 返回一个行集,则调用例程可以循环访问该行集,联接到该行集,或简单地从该行集中选择列。UDF 既类似于视图,又类似于存储过程。像视图一样,UDF 可以返回行集。因此,当 UDF 返回行集并接受参数时,它就像一个可以联接到的存储过程、或者一个参数化的视图。

SQL Server 2000 支持三种用户定义函数:标量函数、内嵌表值函数、多语句表值函数。

\* 标量函数返回在 RETURNS 子句中定义的数据类型的单个数据值。可以使用所有标量数据类型,包括 bigint 和 sql\_variant。不支持 timestamp 数据类型、用户定义数据类型和非标量类型(如 table 或 cursor)。返回类型可以是除 text, ntext, image, cursor 和 timestamp 之外的任何数据类型。

\* 表值函数返回 table。对于内嵌表值函数,没有函数主体;表是单个 SELECT 语句的结果集。

\* 多语句表值函数,在 BEGIN...END 块中定义的函数主体包含 TRANSACT-SQL 语句,这些语句可生成行并将行插入将返回的表中。BEGIN...END 块中的语句不能有任何副作用。函数副作用是指对具有函数外作用域(例如数据库表的修改)的资源状态的任何永久性更改。

函数中的语句唯一能做的更改是对函数上的局部对象(如局部游标或局部变量)的更改。不能在函数中执行的操作包括:对数据库表的修改,对不在函数上的局部游标进行操作,发送电子邮件,尝试修改目录,以及生成返回至用户的结果集<sup>[3]</sup>。

## 2 遍历子树结点的方法

企业或行业应用数据经常存在复杂的树型嵌套关系,比如河流相随机游走储层建模相关的数据就存在很强的这种结构关系。在建模过程中要用到测井数据、小层数据、井位数据等,测井数据又包括孔隙度、饱和度、渗透率等数据,井位数据包括油井三维坐标、井名等。而在进行储层建模数据预处理过程中,需要从不同的数据表中提取相关的子数据,这实际上就是一个树结点遍历的过程。由于相关地质数据量很大,如果采用客户端 SQL 数据请求进行子树节点或者树根节点下的递归遍历,将导致数据库负载增大同时也会引起网络流量的增加和阻塞。因此在遍历算法上采用 SQL Server 数据服务器存储过程或自定义函数方法进行 SQL 编程可以实现高效的数据处理<sup>[4,5]</sup>。文中分别采用存储过程和自定义函数进行遍历算法编程实现。其中关键技术的实现代码为:

\* 树型结构数据的数据存放表:

```
Table MyTreeDATA
([NodeCode] [varchar] (9), -- 结点编码
[NodeName] [varchar] (20), -- 结点名称
[PNodeCode] [varchar] (9) -- 此结点的父结点编码
)
```

\* 遍历子树结点下所有结点的自定义函数:

```
CREATE FUNCTION FindSubTree (@NodeCode varchar
(9)) --
```

```

RETURNS @BasicData table -- 遍历结果表,结构等同与
MyTreeDATA 表
( [NodeCode] [varchar] (9), -- 结点编码
  [NodeName] [varchar] (20), -- 结点名称
  [PNodeCode] [varchar] (9) -- 此结点的父结点编码
) AS
BEGIN
  declare @TempData table(.....) -- 数据临时
  存放表,结构等同与 MyTreeDATA 表
  declare @MTemplate table(.....) -- 中间临时
  表,结构等同与 MyTreeDATA 表
  declare @MExchange table(.....) -- 中间交
  换临时表,结构等同与 MyTreeDATA 表
  delete @TempData, @MTemplate, @MExchange, @BasicData
  insert into @TempData select * from MyTreeDATA
  insert into @MTemplate select * from @TempData where PN-
  odeCode = @NodeCode
  insert into @BasicData select * from @TempData -- 结果
  临时表
  while (exists(select * from @MTemplate where NodeCode in
  (select PNodeCode from @TempData))) -- 当某层的结点全为
  叶子时,停止循环
  begin
    insert into @MExchange select * from @MTemplate --
    中间交换临时表
    delete @MTemplate
    insert into @MTemplate select * from @TempData where
    PNodeCode in (select NodeCode from @MExchange)
    delete @MExchange
    insert into @BasicData select * from @MTemplate
  end
  insert into @BasicData select * from @TempData where
  NodeCode = @NodeCode
  return
END

```

#### \* 存储过程实现:

```

CREATE TABLE #TempData(.....) -- 数据临时
  存放表,结构等同与 MyTreeDATA 表
CREATE TABLE #MTemplate(.....) -- 中间临时
  表,结构等同与 MyTreeDATA 表
CREATE TABLE #MExchange(.....) -- 中间交
  换临时表,结构等同与 MyTreeDATA 表
CREATE TABLE #BasicData(.....) -- 遍历结果
  表,结构等同与 MyTreeDATA 表
CREATE PROCEDURE FindSubTree (@NodeCode varchar

```

```

(9)) -- 求某结点下子树所有结点
AS
truncate table #TempData, #MTemplate, #MExchange, #
BasicData
insert into #TempData select * from MyTreeDATA -- 数
  据临时存放表
insert into #MTemplate select * from #TempData where PN-
  odeCode = @NodeCode -- 中间临时表
insert into #BasicData select * from #MTemplate -- 结果
  临时表
while (exists(select * from #MTemplate where NodeCode in
  (select PNodeCode from #TempData))) -- 当某层结点全为叶
  子时停止循环
begin
  insert into #MExchange select * from #MTemplate --
  中间交换临时表
  truncate table #MTemplate
  insert into #MTemplate select * from #TempData where
  PNodeCode in (select NodeCode from #MExchange)
  truncate table #MExchange
  insert into #BasicData select * from #MTemplate
end
insert into #BasicData select * from #TempData where
NodeCode = @NodeCode
GO

```

### 3 结 语

通过存储过程和自定义函数的遍历实现,并在河流相随机游走储层建模开发中加以实际应用,发现数据提取速度得到明显的改善,特别在多个客户端进行不同数据节点遍历请求时,网络速度快且拥塞现象不明显。

#### 参考文献:

- [1] 陆 鑫.存储过程及其应用方法[J]. 计算机应用,1999,19(2):12-14.
- [2] 万 波,周顺平.SQL SERVER 扩展存储过程实现机制及应用方法初探[J]. 武汉科技大学学报,2001(3):294-297.
- [3] 郑 刚,彭 宏,郑启伦.存储过程在嵌入式多功能数据挖掘器中的应用[J]. 计算机应用,2006,26(6):102-104.
- [4] 王道学.使用 SQL Server 存储过程递归遍历层次结构[J]. 计算机应用,2003,23(6):126-128.
- [5] 檀 明,钟伯成,胡学友,等.一种基于存储过程的 BOM 遍历算法[J]. 合肥学院学报,2007,17(1):40-45.

(上接第94页)

- [5] W3C - The World Wide Web Consortium[EB/OL]. 1994 - 2007. <http://www.w3.org>.
- [6] 李 伟,徐志伟.一种网格资源空间模型及其应用[J]. 计

算机研究与发展,2003,40(12):1756-1762.

- [7] 徐志伟,李晓林,游赣梅.织女星信息网格的体系结构研究[J]. 计算机研究与发展,2002,39(8):948-951.