



[12] 发明专利说明书

专利号 ZL 200480011704.3

[45] 授权公告日 2009 年 9 月 30 日

[11] 授权公告号 CN 100545835C

[22] 申请日 2004.3.31

[21] 申请号 200480011704.3

[30] 优先权

[32] 2003.5.1 [33] US [31] 10/428,443

[86] 国际申请 PCT/US2004/010018 2004.3.31

[87] 国际公布 WO2004/100021 英 2004.11.18

[85] 进入国家阶段日期 2005.10.31

[73] 专利权人 甲骨文国际公司

地址 美国加利福尼亚州

[72] 发明人 拉维·默西

穆拉利达尔·克里希纳普拉萨德

阿南德·马尼库蒂 刘 贞

詹姆士·沃纳

[56] 参考文献

US2001/0037345 A1 2001.11.1

基于 SQL 的 XML 到关系数据库的转换方法. 王亮, 高阳, 陈世福, 陈震解. 计算机应用研究, 第 8 期. 2002

Oracle9i – SQL Reference, Release 2 (9.2), Part No. A96540 – 02. Diana Lorentz, 全文. 2002

Oracle9i – XML Database Developer's Guide – Oracle XML DB, Release 2 (9.2), Part No. A96620 – 02. Shelley Higgins, Part IV Chapter 10, 10.1 ~10.21. 2002

审查员 李 佳

[74] 专利代理机构 北京康信知识产权代理有限责任公司

代理人 余 刚

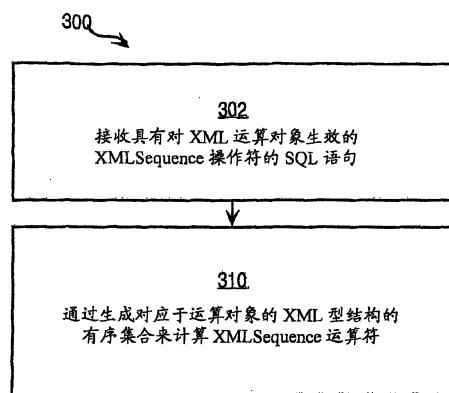
权利要求书 4 页 说明书 28 页 附图 8 页

[54] 发明名称

在 XML 文档和关系数据之间的映射中保留层次信息

[57] 摘要

用于管理在 SQL 兼容 DBMS 中的 XML 数据的方法包括接收 SQL 语句。SQL 语句包括特殊运算符，该运算符对代表第一组 XML 元素的第一 XML 型实体生效。在执行 SQL 语句期间，通过产生 XML 型实体的有序集合来计算特殊运算符。有序集合中的每个不同的实体是基于来自第一组的不同 XML 元素；并且对于来自第一组或来自第一组及其子孙的每个 XML 元素，存在有序集合中的实体。当包括子孙时，有序集合中的每个入口指示在 XML 树中的等级。在另一方面，SQL 语句中的聚集运算符对具有相关等级的实体的集合生效，以产生单一 XML 型实体。



1. 一种用于在允许数据项代表 XML 结构的 SQL 兼容 DBMS 中，在 XML 结构和 SQL 结构之间进行层次数据转换的方法，包括以下步骤：

接收包括特殊运算符的 SQL 语句，所述特殊运算符对至少代表第一 XML 结构的第一数据项生效，所述第一 XML 结构包括第一组一个或多个 XML 结构，对于所述第一组，所述第一 XML 结构是在 XML 树形层次中的祖先节点；以及

在执行所述 SQL 语句期间，通过产生一个或多个入口的有序集合来计算所述特殊运算符，

其中，

所述有序集合中的每个入口包括：

代表所述第一组的特殊 XML 结构的特殊数据项，以及

指示在所述 XML 树形层次中的所述特殊 XML 结构的等级的数据。

2. 根据权利要求 1 所述的方法，其中，指示所述等级的所述数据指示低于所述第一 XML 结构的所述特殊 XML 结构的等级。
3. 根据权利要求 1 所述的方法，所述产生所述有序集合的步骤进一步包括以下步骤：

在 XML 文档顺序中紧接在所述 XML 树形层次的第一节点之后的所述 XML 树形层次的下一节点处，从所述第一组接收当前 XML 结构；

产生当前数据项以代表所述当前 XML 结构；以及
在所述有序集合的入口中，将当前等级与所述当前数据
项相关联。

4. 根据权利要求 3 所述的方法，所述产生所述有序集合的步骤进
一步包括以下步骤：

确定在所述 XML 树形层次中，所述下一节点是否是所述
第一节点的子节点；以及

如果确定所述当前数据项是所述第一节点的子节点，则
通过增加第一等级来产生所述当前等级，所述第一等级与代表
所述第一节点的第二数据项相关联。

5. 根据权利要求 3 所述的方法，所述产生所述有序集合的步骤进
一步包括以下步骤：

确定在所述 XML 树形层次中，所述下一节点是否是所述
第一节点的兄弟节点；以及

如果确定所述下一节点是所述第一节点的兄弟节点，则
产生等于第一等级的所述当前等级，所述第一等级与代表所述
第一节点的第二数据项相关联。

6. 根据权利要求 3 所述的方法，所述产生所述有序集合的步骤进
一步包括以下步骤：

确定在所述 XML 树形层次中，所述下一节点是否是所述
第一节点的祖先节点的兄弟；以及

如果确定所述下一节点是所述祖先节点的兄弟，则通过
减少第一等级一数量来产生所述当前等级，所述第一等级与代表
所述第一节点的第二数据项相关联，所减少的数量与在所述

XML 树形层次中所述下一节点和所述第一节点之间的深度变化有关。

7. 一种用于处理 SQL 语句的方法，所述方法包括：

检测所述 SQL 语句包括特殊运算符，所述特殊运算符指定用于从一个或多个关系表中聚集信息的操作；以及

在执行包括所述特殊运算符的所述 SQL 语句期间，从所述一个或多个关系表中读取属性值、和对应的等级值；以及
产生层次 XML 信息作为所述操作的结果；

其中，在所述层次 XML 信息中，每个所述属性值属于层级级，所述层级级是基于对应于所述属性值的所述等级值。

8. 根据权利要求 7 所述的方法，其中：

所述产生层次 XML 信息的步骤包括将特殊数据项插入到所述结果中；

所述特殊数据项代表 XML 树形层次的下一节点，在 XML 文档顺序中，所述下一节点在所述 XML 树形层次的第一节点之后；

所述第一节点由第一数据项代表；

用于所述第一节点的 XML 结构的结束标记没有被插入到所述结果中；以及

所述第一节点位于所述 XML 树形层次中的第一等级。

9. 根据权利要求 8 所述的方法，所述将所述特殊数据项插入到所述结果中的步骤进一步包括以下步骤：

确定所述特殊数据项的特殊等级是否大于所述第一等级一个增量；以及

如果确定所述特殊等级是大于所述第一等级一个增量，则在插入结束标记之前插入用于该 XML 结构的开始标记。

10. 根据权利要求 8 所述的方法，所述将所述特殊数据项插入到所述结果中的步骤进一步包括以下步骤：

确定所述特殊数据项的特殊等级是否等于所述第一等级；以及

如果确定所述特殊等级等于所述第一等级，则插入结束标记，并且紧接在插入所述结束标记之后，插入用于该 XML 结构的开始标记。

11. 根据权利要求 8 所述的方法，所述将所述特殊数据项插入到所述结果中的步骤进一步包括以下步骤：

确定所述特殊数据项的特殊等级是否小于所述第一等级；以及

如果确定所述特殊等级小于所述第一等级，则插入结束标记，并且减少所述第一等级。

在 XML 文档和关系数据之间 的映射中保留层次信息

技术领域

本发明涉及用于在关系数据库（relational database）系统中使用可扩展标记语言（XML）数据的方法。

背景技术

万维网（WWW）包括互联网上的服务器的网络，每个服务器与一个或多个超文本标记语言（HTML）页面相关联。HTML 页面在对服务器发出请求的客户机和使用超文本传输协议（HTTP）的服务器之间传递。可从互联网上的服务器中获得的资源是用统一资源定位符（URL）定位的。WWW 的标准和协议由万维网联盟（W3C）通过其在 www.w3c.org 的服务器发布，且除在互联网上使用之外，它们还被用于许多专用网路。

HTML 标准是一种更加通用的标记语言标准的一个应用，这种标记语言标准叫做通用标记语言标准（SGML）。最近，SGML 的子集已经被定义，该子集比 HTML 更强大并更灵活，并且已获得普及，用于在互联网及其他网络上传递信息。由 W3C 开发并且促进的新标准叫做可扩展标记语言（XML）。XML 为表示数据中的结构提供了通用语法。结构化数据指的是标记（tag）了其内容、意义、或使用的数据。XML 提供 HTML 中完成的标记（tagging）的扩展，这种扩展集中在格式或表达上。XML 标记识别 XML 元素（element）

及 XML 元素的属性 (attribute)。XML 元素可以被嵌套，以形成元素的层次 (hierarchy)。

假设元素被 XML 定义和使用，则文档对象模型 (DOM) 为树形结构，用来定义在 XML 文档中的信息如何排列。用 XPath 表达式来导航 DOM，该表达式指示在元素的层次中特殊 (particular, 特定) 节点或内容以及 XML 文档中的属性。XPath 是由 W3C 发布的标准。

关系数据库先于万维网且开发独立于万维网。关系数据库将数据存储于对应于数据内的逻辑关系的不同数据容器内。因此，关系数据库支持强大的查询和更新性能。关系数据库通常将数据存储在行和列构成的表中，在表中一行的所有列之间的值是相关的。例如，雇员表一行中的值描述同一雇员的属性，例如她的姓名、社会保险号、地址、薪水、电话号码及其他信息。将每个属性存储在不同的列中。一些称作集合 (collection) 的属性可以有多个入口 (entry, 条目)。例如，雇员可能被允许拥有多个电话号码。特殊结构在一些关系数据库中被定义以存储集合。

关系数据库管理系统 (DBMS) 是在关系数据库中存储和检索数据的系统。关系 DBMS 处理执行数据库功能的请求，例如创建和删除表，增加和删除表中的数据，以及从数据库中的表中检索数据。众所周知的用于表示数据库请求的标准语言是结构化查询语言 (SQL)。

由于作为支持元素间层次关系的数据交换格式的 XML 很流行，以及由于关系 DBMS 的更新和检索数据的能力，需要生成从关系数据库输出的 XML 数据，以及将 XML 数据存储到关系数据库中。在一种方法中，数据库管理员可以委托进行 (commission) 编程工作，以在过程语言中生成代码，该过程语言将特殊 XML 结构

(construct, 构造) 中的数据映射到特殊关系数据库结构中的数据并且返回。这样的编程工作会非常昂贵。

在另一种方法中，可以采用类似于 SQL 语句的说明语句，只表示在 XML 结构和 SQL 结构之间的关系。根据所说明的关系来转换数据的通用程序由 DBMS 供应商一次性编写并且提供给数据库管理员。这使得数据库管理员不用开发过程语言程序来转换数据。为了支持这种需求，已经开发了行业标准 SQL 来对 XML 文档生效 (operate on, 对...起作用)。这种标准叫做 SQL/XML，并且关于 SQL/XML 的信息可在其记录在 www.sqlx.org 上时获得。SQL/XML 提供说明语句，该说明语句可用于只表示在层次 XML 结构中的数据和 SQL 关系结构中的数据之间的一些转换。例如 XMLAgg 是 SQL/XML 函数，其从一组 XML 元素中产生一个 XML 文档，该 XML 元素产生于关系表的选定行。

如这里所用的，XML 结构包括 XML 文档、XML 元素、包括多个 XML 元素的文档片段 (fragment)、以及 XML 元素的 XML 属性等。一种数据，其在 SQL 兼容 DBMS (SQL compliant DBMS) 内被操作，并且由 DBMS 结构化，以支持产生 XML 结构来输送该数据，该数据被称作“XML 型实体 (instance, 实例)”，或者简称为“XML 数据”。这种 XML 数据可能或可能不被存储在如表 (行及列) 这样的 SQL 结构中。

当 SQL/XML 语句为在 XML 结构与 SQL 结构之间的转换中出现的许多情况提供强大工具时，它们不完全适应所有出现的情况。例如，XML 型实体可能包括用于雇员元素的数据，该雇员元素包括几个子元素 (child element)，这些子元素对应于雇员签名离开 (sign out) 的各种装置。DBMS 的用户可能想生成描述满足某种标准的装置的一系列 XML 文档。用于提取那些子元素的传统的 SQL/XML 语句产生关于满足标准的所有装置的单一实体。为了产

生用于每个单独装置的单独 XML 型实体，使用若干传统语句，为每个满足子标准的装置使用一个传统语句。多个语句的生成是冗长的。在一些情况下，不可能预测标准以将每个装置从其他装置中分离。在这样的情况下，能使用单一语句来生成一系列单独 XML 型实体是优选的。

当在 XML 结构和 SQL 结构之间进行转换时，当前的 SQL/XML 语句也不支持保留（retain）一些层次关系。一些 XML 元素可嵌套在其固有类型内以形成层次。例如，叫做“雇员”的 XML 元素可具有一个或多个其它雇员元素作为子元素，子元素代表由第一雇员管理的其他雇员。在一些情况下，如果没有在原始 XML 文档中指示的嵌套，则对文档中若干雇员元素生效的 SQL/XML 函数 EXTRACT 将仅连续地输出所有这些雇员元素。结果是雇员列表没有指示一些雇员是否是被管理员父类雇员（manager parent employee）管理的子雇员（children employee）的层次信息。

一些 SQL 行可能包含表的行内的层次。例如，“emp”表可能包括用于每个雇员的一行并且包括列“mgr”，该列具有指向用于第二雇员的 emp 表的另一行的指针，该第二雇员是第一雇员的管理者。对由这个表的行生成的多个 XML 元素生效的 SQL/XML 函数 XMLAgg 将只产生具有一组根等级（level）元素的 XML 文档片段（例如元素名“雇员”的一系列 XML 元素）。每个雇员元素的 mgr 属性包含了层次；但是，如果雇员元素被嵌套使得所管理的雇员是管理员雇员元素的子元素，则将更令人满意。

根据上文，很明显需要说明语句来加强在 SQL 兼容 DBMS 中的 XML 数据的操作。特别是，当 SQL 结构中的数据被转换为 XML 型实体时，需要说明语句来保留包含在 SQL 结构内的层次关系。当实体被转换为多个 XML 型实体时，也特别需要说明语句来保留 XML 型实体里的层次关系。

可以继续本节中所描述的方法，但不是已被构思和继续的必要的方法。因此，除非在此另有说明，在此部分中描述的方法不能仅因为在此背景技术部分中出现这些方法，就认为是本申请的权利要求中的现有技术。

附图说明

本发明在附图中以举例而不是限制的方式加以阐明，附图中相同的附图标记指示相同的元件，其中：

图 1 是示出根据实施例的 XML 文档、XML 图解、以及将 XML 文档内容存储于对象-关系数据库内的对象-关系结构的框图；

图 2A 是示出树形层次的一个实例的框图；

图 2B 是示出在对应于图 2A 的树形层次的 XML 文档实例的 201 部分中的 XML 结构的框图；

图 3 是示出根据实施例的用于产生 XML 型 SQL 结构的集合的方法的概述的流程图；

图 4A 是示出根据实施例的用于产生具有附加 (additional) 层次信息的 XML 型 SQL 结构的集合的方法的概述的流程图；

图 4B 是示出图 4A 的方法的步骤的实施例的流程图；

图 5A 是示出根据实施例的聚集 (aggregate) 具有层次信息的 XML 型 SQL 结构的集合的方法的概述的流程图；

图 5B 是示出图 5A 的方法的步骤的实施例的流程图；以及

图 6 是示出计算机系统的框图，可在该系统上执行本发明的实施例。

具体实施方式

描述了用于操作在 SQL 兼容 DBMS 中的 XML 数据的技术。在下面的描述中，出于解释的目的，许多具体的细节被阐明以提供对本发明的彻底理解。然而，显而易见，本发明可以在没有这些具体细节的情况下实施。在其它实例中，已知的结构和装置以框图的形式示出，以避免对本发明造成不必要的混淆。

功能概述

提供了新的 XML 操作用于操作在 SQL 兼容 DBMS 中的 XML 数据。DBMS 允许 XML 型实体代表 XML 结构，例如 XML 文档、XML 元素、包括若干 XML 元素的 XML 文档片段、以及 XML 元素的 XML 属性。根据本发明的一个方面，本技术包括在 DBMS 中接收 SQL 语句。SQL 语句包括对代表第一组零个或多个 XML 元素的第一 XML 型实体生效的顺序运算符 (operator)。在执行 SQL 语句期间，通过生成零个或多个 XML 型实体的有序 (ordered) 集合来计算 (evaluate, 求值) 顺序运算符。有序集合中的每个不同的 XML 型实体都基于来自第一组的不同 XML 元素，并且对于在第一组中的每个 XML 元素，有序集合中存在 XML 型实体。这些技术允许说明语句加强对 XML 型 SQL 实体的操作。

根据本发明的另一方面，接收包括层次顺序运算符的 SQL 语句。层次顺序运算符对代表至少第一 XML 结构的第一 XML 型实体生效，该至少第一 XML 结构包括第一组一个或多个 XML 结构，其中第一 XML 结构是在 XML 树形层次中的祖先 (ancestor) 节点。在执行 SQL 语句期间，通过生成一个或多个入口的有序集合来计算

层次顺序运算符。有序集合中每个入口都包括特殊 XML 型实体和指示等级的数据。特殊 XML 型实体代表第一组中的特殊 XML 结构。数据指示 XML 树中特殊 XML 结构的等级。当那些 XML 结构被作为多个 XML 型实体进行操作时，本方面的技术允许说明语句保留在 XML 结构中的层次关系。

根据本发明的另一方面，接收包括层次聚集运算符的 SQL 语句。层次聚集运算符对特殊 XML 型实体和指示特殊等级的数据生效。特殊 XML 型实体是将被包括在代表祖先 XML 结构的结果实体中的多个 XML 型实体之一。祖先 XML 结构具有相关的 XML 树形层次。特殊等级指示 XML 树形层次中将放置由特殊 XML 型实体代表的特殊 XML 结构的地方的等级。在执行 SQL 语句期间，通过将特殊 XML 型实体以由等级指示的深度（depth）插入到结果 XML 型实体中来计算层次聚集运算符。这些技术允许当那些 XML 结构的数据被转换成 XML 型实体或者以 XML 结构输出时，说明语句保留包含在 SQL 结构中的层次关系。

下面描述关于商业可扩展标记语言/结构化查询语言（XML/SQL）数据库服务器的实施例，该服务器使用对象-关系结构以存储一个或多个 XML 文档的内容，并且响应对于该内容的 XPath 查询，该内容将被当作一个或多个 XML 文档或者由一个或多个 XML 结构组成的文档片段来操作或输出。XPath 查询包括用于定位在 XML 文档中的 XML 文件的 XPath 表达式，以及零个或多个 SQL/XML 函数，用于生成、合并或比较被 XPath 表达式定位的数据。SQL/XML 标准函数包括 EXTRACT、EXTRACTVALUE、以及 EXISTSNODE，这些对于本领域技术人员而言是众所周知的。但是本发明不局限于这个范围，还可能适用于包括关系数据库结构和 XML 数据的任何范围。

结构概述

包括在本发明实施例中的结构包括具有 XML 结构和关系数据库结构（也称作 SQL 结构）的 XML 文档。图 1 是示出 XML 文档实例 110，以及对象-关系结构实例的框图，该结构将 XML 文档内容存储于由对象-关系数据库服务器 130 管理的对象-关系数据库中。

XML 文档 110 是特殊 XML 文档型实体，在下文中被称作“ORG”XML 文档，它描述企业内雇员间的组织关系。为了进行说明，假设 ORG 文档实例 110 包括用于公司管理人员的类型 EMPLOYEE（名为“EMPLOYEE”）的 XML 元素 112a。每个 EMPLOYEE 型元素包括雇员编号（名为“ENO”）的属性（未示出），以及类型 ENAME、EINFO 的两个 XML 元素，以及类型 EMPLOYEE 的零个或多个其他元素。类型 ENAME 的 XML 元素，例如 XML 元素 114a 存有指示雇员名称的内容。类型 EINFO 的 XML 元素，例如 XML 元素 114b 存有指示其他雇员信息（例如地址和薪水）的内容。所包括的类型 EMPLOYEE 的 XML 元素，例如 XML 元素 116a、116b 存有指示由通过属性和 ENAME 及 EINFO 元素的内容所识别的雇员所管理的雇员。EMPLOYEE 型元素，例如 XML 元素 116a、116b 和由省略符号 115 指示的其它元素，在下文中被共同作为子 EMPLOYEE 元素 116。

XML 数据库服务器实例 130 是 XML 对象-关系数据库服务器，其导入和导出 XML 文档，该文档作为 XML 型实体（在这里也称作 XML 实体）代表一个或多个 XML 结构，并且将 XML 结构的内容存储在数据库存储空间 140 中的一个或多个 SQL 结构中。数据库存储空间 140 包括一个或多个其它 SQL 结构，例如表 144 和图 148。

在所述的实施例中，表并不直接涉及 XML ORG 文档。例如，不存在每行都对应不同 ORG 文档或不同 EMPLOYEE 元素的表。使上述情况可能出现的一种方法是，在 ORG 型 XML 文档被定义之前，数据可能已经被存储在 EMP 表 144 中，并且用于多个应用。另一种情况是 EMP 故意被设计为比 XML 型实体的表更加紧凑。EMP 表并不是 XML 型表；并且 EMP 表包括具有列名称 mgr 的列以存储代表雇员的管理人员的数据，该列并不是 XML 型列。

对于例如非 XML 型的表 144 和列 MGR 的对象关系结构，XML 查询没有意义。必须产生 XML 型对象用于这种 XML 查询。通过使用在 SQL/XML 中的一个或多个 XML 生成函数，XML 型对象可在 XML 对象-关系数据库服务器中从不是 XML 型实体的 SQL 结构中产生。例如，XML 生成函数 XMLElement 从一个或多个非 XML 型的对象或者标量列中产生 XML 型实体。XML 型对象（XML 型实体）可被 DBMS 使用以产生 XML 结构的数据流。XML 生成函数 XMLAttributes 从标量列中产生 XML 型对象的属性。也可使用一个或多个 SQL/XML 函数结合 XML 型对象。包括 XML 生成函数的子查询必须包括在指向非 XML 型的对象-关系结构中的数据的 XML 查询中。在一些情况下，包括 XML 生成函数的子查询可能被储存为 XML 类型图，例如 XML 类型图 148。

新 XML 函数实例

根据一些实施例，三个用于加强在 SQL 兼容 DBMS 中的 XML 数据的管理的 XML 函数实例，被称作 XMLSequence、HierXMLSequence、以及 HierXMLAggregate，它们在下文中会有更详细说明。

XML 结构和 SQL 结构的内容实例

为了说明在说明 SQL 语句中的用于操作 XML 数据的这些函数的使用，假设列于表 1 中的数据存在于 EMP 表 144 中。

表 1. EMP 表 144 的内容实例

ENO	ENAME	EINFO	MGR
1	Linda	...	null
2	Charles	...	1
3	Terry	...	1
4	Alice	...	2
5	Mary	...	2
6	Ray	...	4
7	Vishnu	...	6
8	Cetin	...	6
9	Steve	...	3

该 EMP 表是使用 SQL 结构的可交换行的平面文件；但是包含雇员的层次。例如，列于表 1 中的内容指示 Linda (ENO=1) 为层次的最高层；并且 Charles (ENO=2) 和 Terry (ENO=3) 向她汇报。Alice (ENO=4) 和 Mary (ENO=5) 向 Charles 汇报。Ray (ENO=6) 向 Alice 汇报。Vishnu (ENO=7) 和 Cetin (ENO=8) 都向 Ray 汇报。Steve (ENO=9) 向 Terry 汇报。这些包含的层次在图 2A 中作为节点树示出。

图 2A 是示出树形层次的框图。在图 2A 中，节点 205 包括节点 210、220、230、240、250、260、270、280、290。根节点 210 代表雇员 Linda。Charles 和 Terry 两个雇员，分别由子节点 (child node) 220、230 代表，向由节点 210 代表的 Linda 汇报。代表 Terry 的节点 230 有一个子节点 290，代表向 Terry 汇报的雇员 Steve。代表 Charles 的节点 220 有两个子节点 240、250，分别代表向 Charles 汇报的雇员 Alice、Merry。代表 Alice 的节点 240 有一个子节点 260，代表向 Alice 汇报的雇员 Ray。代表 Ray 的节点 260 有两个子节点 270、280，分别代表向 Ray 汇报的雇员 Vishnu、Cetin。

图 2B 示出用 XML 结构表示相同信息的 XML ORG 文档。图 2B 是示出在 XML 文档实例的部分 201 中的 XML 结构的框图。在图 2B 中，应用 XML 结构示出 ORG 型 XML 文档的部分 201。部分 201 中的每一行都由列于图 2B 中行左边的行号 202 指示。第 1 行中的省略符号表示先于文档中的根元素 ORG 的 XML 结构，例如指示 XML 版本以及将用于有效元素和属性的名字空间的数据。在节点 210，代表 Linda 的 XML 结构包括第 4 行到 29 行中的那些结构。代表 Linda 的子孙元素 (descendent element) 的 XML 结构由标记 220 到 290 的括弧指示，用于图 2A 中的对应节点。尽管为了更清楚地讨论，将 XML 结构分开到不同的行上，XML 并没有采用换行符来区别 XML 结构。图 2A 的树形层次 204 与图 2B 中的 ORG XML 文档的数据对象模型 (DOM) 相对应。

值得注意的是，mgr 列不必与任何 XML 结构相对应——管理人员-被管理雇员关系由嵌套的 XML 元素的父-子 (parent-child) 关系指示。同样值得注意的是，数据的 XML 表示比在表 1 中所示的 EMP 表中相同数据的表示更为冗长。

顺序运算符

顺序运算符用于从 XML 文档或片段中产生 XML 型子实体 (child instance) 的集合。为了说明的目的，假设将 ELIST 文档存储在文件 ELIST.xml 中，并且包括雇员列表，以及如表 2a 所示，被表示为 XML 型实体。进一步假设期望它作为单独 XML 型实体操作 EMPLOYEE XML 元素。

表 2a. ELIST 元素

行	ELIST 元素
1	<ELIST>
2	<EMPLOYEE ENO = "1">
3	<ENAME> Linda </ENAME> <EINFO> ... </EINFO>
4	</EMPLOYEE>

5	<EMPLOYEE ENO = "2">
6	<ENAME> Charles </ENAME> <EINFO> ... </EINFO>
7	</EMPLOYEE>
8	<EMPLOYEE ENO = "4">
9	<ENAME> Alice </ENAME> <EINFO> ... </EINFO>
10	</EMPLOYEE>
11	<EMPLOYEE ENO = "6">
12	<ENAME> Ray </ENAME> <EINFO> ... </EINFO>
13	</EMPLOYEE>
14	<EMPLOYEE ENO = "7">
15	<ENAME> Vishnu </ENAME> <EINFO> ... </EINFO>
16	</EMPLOYEE>
17	<EMPLOYEE ENO = "8">
18	<ENAME> Cetin </ENAME> <EINFO> ... </EINFO>
19	</EMPLOYEE>
20	<EMPLOYEE ENO = "5">
21	<ENAME> Mary </ENAME> <EINFO> ... </EINFO>
22	</EMPLOYEE>
23	<EMPLOYEE ENO = "3">
24	<ENAME> Terry </ENAME> <EINFO> ... </EINFO>
25	</EMPLOYEE>
26	<EMPLOYEE ENO = "9">
27	<ENAME> Steve </ENAME> <EINFO> ... </EINFO>
28	</EMPLOYEE>
29	</ELIST>

标准 SQL/XML 函数 EXTRACT 可以产生列出 EMPLOYEE XML 元素的 XML 型实体。例如，命令

EXTRACT ('ELIST.xml', 'ELIST/EMPLOYEE')

从 ELIST.xml 文档中提取 XML 元素 EMPLOYEE，并且生成表 2b 所示的代表 XML 结构的结果 XML 型实体。在这个实体中，所有雇员都是第一代 (generation) 子 XML 元素。例如，雇员 Linda 由第 1-3 行的元素表示，雇员 Charles 由第 4-6 行的元素表示。由于不存在单一根元素，但是在根等级上存在多个 XML 元素，因此该实体代表片段而不是 XML 文档。

表 2b. 用 EXTRACT 命令实例生成的 XML 型实体实例

行	来自 EXTRACT 函数的 XML 实体
1	<EMPLOYEE ENO = "1">
2	<ENAME> Linda </ENAME> <EINFO> ... </EINFO>
3	</EMPLOYEE>
4	<EMPLOYEE ENO = "2">

```

5   <ENAME> Charles </ENAME> <EINFO> ... </EINFO>
6   </EMPLOYEE>
7   <EMPLOYEE ENO = "4">
8     <ENAME> Alice </ENAME> <EINFO> ... </EINFO>
9   </EMPLOYEE>
10  <EMPLOYEE ENO = "6">
11    <ENAME> Ray </ENAME> <EINFO> ... </EINFO>
12  </EMPLOYEE>
13  <EMPLOYEE ENO = "7">
14    <ENAME> Vishnu </ENAME> <EINFO> ... </EINFO>
15  </EMPLOYEE>
16  <EMPLOYEE ENO = "8">
17    <ENAME> Cetin </ENAME> <EINFO> ... </EINFO>
18  </EMPLOYEE>
19  <EMPLOYEE ENO = "5">
20    <ENAME> Mary </ENAME> <EINFO> ... </EINFO>
21  </EMPLOYEE>
22  <EMPLOYEE ENO = "3">
23    <ENAME> Terry </ENAME> <EINFO> ... </EINFO>
24  </EMPLOYEE>
25  <EMPLOYEE ENO = "9">
26    <ENAME> Steve </ENAME> <EINFO> ... </EINFO>
27  </EMPLOYEE>

```

为了对于每个雇员生成单独 XML 型实体，期望可以从表 2b 中所示的输出文档中产生 XML 型实体的序列。XML 型实体的序列可以当做 XML 型实体的源，用于在表中存储或在 FROM 子句中临时使用。需要顺序函数以产生 XML 型实体的集合。例如，在表 3a 中描述了从表 2b 的输出文档中产生的 XML 型实体的集合。这些实例按与输出的 XML 文档中的那些元素的顺序相对应的顺序出现。

表 3a. 来自 XMLSequence 的 XML 型的序列实例

实体 #	XML 实体
1	<EMPLOYEE ENO = "1"> <ENAME> Linda </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
2	<EMPLOYEE ENO = "2"> <ENAME> Charles </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
3	<EMPLOYEE ENO = "4"> <ENAME> Alice </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
4	<EMPLOYEE ENO = "6">

	<ENAME> Ray </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
5	<EMPLOYEE ENO = "7"> <ENAME> Vishnu </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
6	<EMPLOYEE ENO = "8"> <ENAME> Cetin </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
7	<EMPLOYEE ENO = "5"> <ENAME> Mary </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
8	<EMPLOYEE ENO = "3"> <ENAME> Terry </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
9	<EMPLOYEE ENO = "9"> <ENAME> Steve </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>

根据本发明该方面的实施例，设置了 XMLSequence 运算符，该运算符对代表 XML 结构的 XML 型实体生效，如表 2b 所示，并且产生 XML 型实体的文档有序集合，如表 3a 所示。

图 3 是示出根据实施例的产生 XML 型实体的集合的方法 300 的概述的流程图。尽管在图 3 和随后的流程图中，以特殊的顺序示出步骤，但在其他实施例中，步骤可能以不同顺序执行或者时间上有重叠。

在步骤 302 中，接收具有 XMLSequence 运算符的 SQL 语句，该运算符对代表 XML 结构的 XML 型运算对象生效。例如，接收如下所示 SQL 语句 S1a:

```
SELECT * FROM
  TABLE (
    XMLSequence(EXTRACT('ELIST.xml','ELIST/EMPLOYEE'))
  )
```

S1a.

在步骤 310 中，通过生成包括 XML 型实体的有序集合来计算 XMLSequence 运算符，该 XML 型实体与运算对象的子节点相对应。例如，XMLSequence 对运算对象 EXTRACT ('ELIST.xml'，

‘ELIST/EMPLOYEE’) 生效，其生成表 2b 中的 XML 结构。通过生成具有表 3a 中所示的 XML 型实体的有序集合来计算 XMLSequence，表 3a 中所示的 XML 型实体与表 2b 所示的运算对象的子节点相对应。语句 S1 中的运算对象是从 EXTRACT 函数输出的 XML 型实体。其他实施例中，运算对象可能是 XML 文件或者存有 XML 数据的其他数据结构。

可以看到，这种方法提供了用于说明语句中的 XMLSequence 运算符，其加强了在 SQL 兼容 DBMS 中的 XML 数据的操作。

不需要使运算对象的所有子节点成为与上述实例中相同类型的 XML 元素。例如，对 XML 元素 EMPLOYEE 生效的 XMLSequence 将生成两个 XML 型实体，一个代表 XML 元素 ENAME，第二个代表 XML 元素 EINFO。

使用 XMLSequence，只有满足一定标准的子元素才可被当作单独实体轻易地输出。例如，语句 S1a 中 XPath 表达式可以修改为语句 S1b，以包括指示 ENO 小于 4 的判定 “[ENO<4]”。

```
SELECT * FROM
  TABLE (
    XMLSequence(
      EXTRACT('ELIST.xml','ELIST/EMPLOYEE[ENO<4]')
    )
  ) S1b.
```

此修改生成列于表 3b 中的 XML 型实体的集合。

表 3b. 来自 XMLSequence 的 XML 型实体的序列实例

实体 #	XML 实体
1	<EMPLOYEE ENO = "1"> <ENAME> Linda </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
2	<EMPLOYEE ENO = "2"> <ENAME> Charles </ENAME> <EINFO> ... </EINFO>

	</EMPLOYEE>
3	<EMPLOYEE ENO = "3"> <ENAME> Terry </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>

在另一应用实例中，根据语句 S1c，EXTRACTVALUE 函数可用于提取具有雇员编号的给定范围的雇员的姓名。

```
SELECT EXTRACTVALUE (value(t),'/EMPLOYEE[ENO<4]/ENAME')
    FROM TABLE (
        XMLSequence(EXTRACT('ELIST.xml','ELIST/EMPLOYEE'))
    ) t
```

S1c.

此修改生成列于表 3c 总的文本值的集合。

表 3c. 来自关于 XMLSequence 输出的 EXTRACTVALUE 的文本值

文本#	文本值
1	Linda
2	Charles
3	Terry

层次顺序运算符

在一些情况下，可能期望序列不丢弃 XML 结构中的层次信息。XMLSequence 不适用于在所有情况下的这个目的。为了说明的目的，假设 ORG 文档被存储在文件 ORG.xml 中，并且假设期望在 ORG 文档内产生雇员列表以及期望操作作为单独 XML 型实体的雇员。例如，如果 XMLSequence 直接对 ORG.xml 中存储的 XML 文档生效，则只产生对应于根节点 Linda 的具有所有其子孙（descendent）的一行。

图 4A 是示出根据实施例的用于产生具有附加层次信息的 XML 型 SQL 结构集合的方法 400 的概述的流程图。

在步骤 402 中，接收具有 HierXMLSequence 运算符的 SQL 语句，该运算符对 XML 运算对象生效。例如，接收如下所示 SQL 语句 S2：

```
SELECT * FROM
  TABLE (HierXMLSequence('ORG.xml'))
```

S2.

在步骤 410 中，通过生成有序集合来计算 HierXMLSequence 运算符，该有序集合包括对应于运算对象的所有子孙节点 (descendent node) 的 XML 型实体以及包括树形层次中每个子孙节点的等级。例如，HierXMLSequence 对图 2B 所示的运算对象 ‘ORG.xml’ 生效。通过生成表 4 所示的有序集合来计算 HierXMLSequence，该有序集合具有层次中的等级和对应于图 2B 所示的运算对象的子孙节点的 XML 型实体。在一些实施例中，包括另一个运算对象以指示对应于包括在集合中的 XML 型实体的元素类型，如 EMPLOYEE。

表 4. 来自 HierXMLSequence 的 SQL 对象类型的序列实例

对象#	等级 编号	XML 实体
1	1	<EMPLOYEE ENO = "1"> <ENAME> Linda </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
2	2	<EMPLOYEE ENO = "2"> <ENAME> Charles </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
3	3	<EMPLOYEE ENO = "4"> <ENAME> Alice </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
4	4	<EMPLOYEE ENO = "6"> <ENAME> Ray </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
5	5	<EMPLOYEE ENO = "7"> <ENAME> Vishnu </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
6	5	<EMPLOYEE ENO = "8"> <ENAME> Cetin </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
7	3	<EMPLOYEE ENO = "5"> <ENAME> Mary </ENAME> <EINFO> ... </EINFO>

		</EMPLOYEE>
8	2	<EMPLOYEE ENO = "3"> <ENAME> Terry </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>
9	3	<EMPLOYEE ENO = "9"> <ENAME> Steve </ENAME> <EINFO> ... </EINFO> </EMPLOYEE>

可以看到，这种方法提供了用于说明语句的 HierXMLSequence 运算符，当 XML 结构被转换为 XML 型实体的集合时，该运算符保留在 XML 结构中的层次关系。

图 4B 是示出图 4A 方法的步骤 410 的实施例的流程图。在步骤 412 中，对一些变量进行初始化。指示当前等级的变量用值 0 进行初始化。在所示的实施例中，指示等级的数据在根等级上具有值 1，并且每沿树向下一代其值加 1。在其他实施例中，等级可以用其他数据表示，例如比 1 大的增量、质数、字母表的字母、或者存储的预定数字的序列。指示前一节点的变量被设置为空值，以指示没有树节点已被处理。

在步骤 414 中，树中的下一节点的下一 XML 结构按照文档顺序被接收。例如，用于开始代表 Linda 的元素的 XML 结构被接受，包括来自图 2B 中第 3 行的具有属性 “ENO=1” 的标记 “<EMPLOYEE...>”。在所描述的实施例中，还接收来自图 2B 的第 4 行的子元素 ENAME 和 EINFO。

在步骤 420 中，确定下一 XML 结构是否是前一节点的子节点或者前一节点是否为空。例如，如果在用于对应于前一节点的元素的结束标记之前遇到用于下一 EMPLOYEE 元素的起始标记，则下一元素是一个子节点。如果是，那么控制转到步骤 422。如果不是，那么控制转到步骤 430。

在步骤 422 中，增加当前等级，并且产生有序集合的行。例如，当前等级从 0 变为 1，并且生成表 4 中代表实体的集合的第一元素的第一行，其具有子元素 ENAME 和 EINFO 的内容。然后，控制转到步骤 450。

在步骤 450 中，重置指示前一节点的变量，以指示当前的下一分点。然后，控制转到步骤 414 来接收新的下一分点（如果有的话）。例如，在步骤 422 期间处理起始标记和节点 210 的元素之后，在步骤 450 期间节点 210 成为前一节点。

在所述实施例中，ORG.xml 文档中的下四个 XML 元素全部为前一节点的子元素，因此，对于每个元素都重复步骤 420、422、450。每次将等级增加到 2、然后是 3、4 和 5。

在步骤 430 中，确定下一节点是否是前一节点的兄弟（ sibling）。例如，如果在用于对应于前一节点的元素的结束标记之后并且在任何其他结束标记之前遇到用于下一 EMPLOYEE 元素的起始标记，则下一元素是兄弟节点。如果下一元素是兄弟节点，则控制转到步骤 432。如果不是，则控制转到步骤 440。

在步骤 432 中，当前等级没有发生变化，并且产生有序集合的另一元素。例如，当在代表 Vishnu 的节点 270 之后遇到代表 Cetin 的兄弟节点 280 时，当前等级为 5。等级值没有变化，并且生成表 4 的第 6 行用于代表 Cetin 的节点，其具有子元素 ENAME 和 EINFO 的内容。然后控制转到步骤 450，如上所述，重置指示前一节点的变量并且回到步骤 414，以接收任意附加的 XML 结构。

在所示的实施例中，起始于第 19 行并且代表用于 Mary 的节点 250 的文档 ORG.xml 中的下一 XML 元素，既不是用于 Cetin 的前一节点 280 的子女（ child）也不是它的兄弟，因此控制转到步骤 440。

在步骤 440 中，确定下一节点是否是前一节点的祖先或是前一节点的祖先的兄弟。例如，如果在用于对应于前一节点元素的结束标记之后并且在任何附加的结束标记之后遇到用于下一 EMPLOYEE 元素的起始标记，则下一元素是祖先或祖先的兄弟。如果下一元素是祖先或祖先的兄弟，则控制转到步骤 442。如果不是，则指示错误，并且控制转到步骤 490 以处理该错误。

在步骤 442 中，当前等级减少与附加结束标记的数量相关的数值，在用于下一节点的 XML 结构的起始标记之前遇到该附加结束标记。产生有序集合的另一元素，具有适当减少的等级和 XML 型实体。例如，当在代表 Cetin 的节点 280 之后遇到代表 Mary 的祖先兄弟节点 250 时，两个附加的雇员结束标记相遇（在图 2B 中的第 17 行和 18 行），因此将当前等级减 2，从旧值 5 减到新值 3。产生表 4 的第 7 行用于代表 Mary 的节点，具有子元素 ENAME 和 EINFO 的内容。然后控制转到步骤 450，如上所述，重置指示前一节点的变量并且返回到步骤 414 以接收任意附加的 XML 结构。

以该方式继续，从 XML 运算对象 ORG.xml 中产生表 4 的 XML 型实体和相关等级。

层次聚集运算符

在一些情况下，将 XML 型实体的集合合并成代表祖先 XML 结构的单一实体。标准 XMLAgg 函数用于产生 XML 实体，该实体合并多个 XML 类型的 XML 输出。在聚集 XML 实体中，所有从集合中产生的 XML 实体在聚集 XML 实体中表现为兄弟。例如，可以用 XMLAgg 来聚集表 3a 的集合中的 XML 类型，以产生表 2b 中所述的 XML 实体。

在一些情况下，可期望聚集 XML 实体合并与将被聚集的 XML 型实体相关的层次信息。例如，表 4 中的等级信息应该被用于生成图 2B 中所示的 XML ORG 文档中的嵌套的 XML 元素。

图 5A 是示出根据实施例的用于聚集具有层次信息的 XML 型实体的集合的方法 500 的概述的流程图。

在步骤 502 中，接收具有 HierXMLAggregate 运算符的 SQL 语句，该运算符对集合中的 XML 型运算对象和等级生效。为了说明的目的，假设表 4 的行可从叫做 VEMP 的 XML 型图中得到。在后面的部分中，将对从 EMP 表中生成 VEMP 的查询进行更详细地描述。例如，接收如下所示 SQL 语句 S3：

```
SELECT HierXMLAggregate(LEVEL, EMPLOYEE) FROM VEMP           S3.
```

在步骤 510 中，通过将下一 XML 型运算对象在用于 XML 结构的层次的等级处插入到代表祖先 XML 结构的 XML 型实体中，来计算 HierXMLAggregate 运算符，该等级取决于等级运算对象。例如，通过将下一 EMPLOYEE 在基于 LEVEL 值的层次的等级处插入到临时输出 XML 文档中，HierXMLAggregate 对来自 VEMP 的 XML 型运算对象 EMPLOYEE 和来自 VEMP 的运算对象 LEVEL 生效。输出的 XML 文档或片段对应于图 2B 的第 3 行到 29 行。

可以看到，本方法提供了用于说明语句的 HierXMLAggregate 运算符，当 XML 型实体被合并成代表 XML 结构的单一实体时，该运算符保留与那些实体有关的层次信息。

图 5B 是示出图 5A 方法的步骤 510 的实施例的流程图。在步骤 512 中，将一些变量初始化。指示前一等级的变量用值 0 初始化。指示前一节点的变量被设置为空值，以指示没有树节点已被处理。

在步骤 514 中，按照文档顺序接受下一 XML 型实体和树中的相关等级。例如，代表 Linda 的 XML 型 EMPLOYEE 实体，包括用于 ENO、ENAME 和 EINFO 的值，从表 4 的第 1 行接收该实体和相关等级 1。相关等级的值储存在代表当前等级的变量中。

在步骤 520 中，确定当前等级是否等于增加了一个等级的前一等级。具有满足此条件的等级的下一 XML 型实体代表前一节点的子节点。如果是，则控制转到步骤 522。如果不是，则控制转到步骤 530。例如，当前等级 1 等于前一等级 0 加 1，因此控制转到步骤 522。

在步骤 522 中，下一 XML 型实体作为前一结构的子女被插入。例如，在插入用于前一节点的前一 XML 结构的结束标记之前插入用于下一 XML 结构的起始标记。在所述的实施例中，对应于 XML 型 EMPLOYEE（包括属性 ENO 的值）的 XML 结构 EMPLOYEE 的起始标记被增加到代表 XML 文档或片段的 XML 型输出中。在所述的实施例中，起始和结束标记以及 ENAME 和 EINFO 元素的值也被插入到该输出中。因此，对应于图 2B 的第 3 和第 4 行的行被插入到输出中。然后控制转到步骤 550。

在步骤 550 中，指示前一节点的变量被重置，用于指示当前的下一节点。然后控制转到步骤 552。

在步骤 552 中，确定将被聚集的最后一 XML 型实体是否已被接收。如果不是，则控制转回到步骤 514 以接受新的下一 XML 型实体。如果用于聚集的最后一 XML 型实体已经被接收，则控制转到步骤 554，以终止输出的产生，这将在随后的部分中进行更详细描述。

在所述的实施例中，表 4 中的下四个 XML 型实体都将它们的等级增加 1，达到值 2，然后是 3、4 和 5，指示所有都代表前一节点的子节点，因此为每个实体重复步骤 520、522、550、552。因此，将对应于图 2B 中的第 5-12 行的行插入到输出中。

在步骤 530 中，确定当前等级是否等于前一等级。具有满足该条件的等级的下一 XML 型实体代表前一节点的兄弟节点。如果是，则控制转到步骤 532。如果不是，则控制转到步骤 540。例如，当对表 4 中用于雇员 Cetin 的第 6 行生效时，当前等级 5 等于前一等级 5，因此控制转到步骤 532。

在步骤 532 中，下一 XML 型实体作为前一结构的兄弟被插入。例如，在插入用于下一 XML 结构的起始标记之前，插入用于前一节点的前一 XML 结构的结束标记。在所述的实施例中，在用于雇员 Cetin 的 XML 结构 EMPLOYEE 的起始标记增加给输出之前，插入用于 XML 结构 EMPLOYEE 的结束标记。在所述的实施例中，用于 EMPLOYEE 的起始标记包括属性 ENO 的值；并且起始和结束标记以及 ENAME 和 EINFO 元素的值也被插入到输出的 XML 文档中。因此，对应于图 2B 的第 13-15 行的行被插入到输出中。然后控制转到步骤 550。

在步骤 540 中，确定当前等级是否小于前一等级。具有满足该条件的等级的下一 XML 型实体代表前一节点的祖先或祖先节点的兄弟。如果当前等级较小，则控制转到步骤 542。如果不是，则控制转到步骤 590。例如，当对表 4 中用于雇员 Mary 的第 7 行生效时，当前等级 3 小于前一等级 5，因此控制转到步骤 542。

在步骤 542 中，插入用于一个或多个前一 XML 结构的结束标记。在所述的实施例中，只要当前等级小于前一等级，就插入用于 XML 结构 EMPLOYEE 的结束标记并且降低前一等级。例如，当对

表 4 用于雇员 Mary 的第 7 行生效时，当前等级为 3 时，插入两个结束标记，且降低前一等级两次，从 5 到 4 再到 3。因此，将对应于图 2B 中的第 16-17 行的行插入到输出中，且前一等级被设置为 3。在所述的实施例中，在当前等级不小于前一等级时，则控制转到步骤 514。在其它的实施例中，在当前等级不小于前一等级时，控制可转到步骤 520 或 530 或 532。因此，将对应于图 2B 的第 18-20 行的行插入到输出中。

当试图产生低于当前等级的两代或更多代的子孙时，控制转到步骤 590。这种在等级上的变化在 XML 文档中无效。在步骤 590 中，处理错误。例如，终止产生 XML 结构，并且将警告信息发给数据库的用户。

以该方法继续，第 3 到 26 行的 XML 结构从来自表 4 的 XML 型运算对象中产生。在插入第 26 行之后，在步骤 552 中确定表 4 中不再有 XML 型实体。控制转到步骤 554。

在步骤 554 中，插入用于一个或多个前一 XML 结构的结束标记。在所述的实施例中，只要当前等级小于 0，就插入用于 XML 结构 EMPLOYEE 的结束标记并且降低前一等级。例如，在对表 4 用于雇员 Steve 的第 9 行生效之后，当前等级为 3。因此插入三个结束标记并且降低前一等级三次，从 3 到 0。将对应于图 2B 的第 27-29 行的行插入到输出的 XML 文件中。

产生用于聚集运算对象的等级

使用 HierXMLAggregate 运算符需要等级运算对象。可以用 SQL“CONNECT BY”子句从类似表 1 的具有包含的层次的 SQL 表中产生等级运算对象的值。例如，具有表 4 所示的等级的 SQL 图 VEMP 可以从表 1 所示的 EMP 表中产生，使用下列 SQL 语句 S4:

```

CREATEVIEW VEMP AS
    SELECT level,
        XMLElement("EMPLOYEE", XMLAttribute(eno), e.ename, e.einfo)
    AS EMPLOYEE
    FROM EMP e
    CONNECT BY prior e.eno = e.mgr
    START WITH e.mgr = null

```

S4.

在上述语句中，CONNECT BY 子句产生伪列“等级”，以指示所产生的层次中的等级。

层次从 EMP 的行开始，EMP 中 MGR 列中的值为空。下一行是表中的下一行，表中 MGR 列中的值等于前一行的 ENO 列中的值。该过程按预订深度第一顺序产生一组行，该顺序等于 XML 文档顺序。例如，用于父代的行在其任何子女的行产生之前产生，并且用于一个兄弟的所有叶节点的行在产生用于下一兄弟和其任何子孙的行之前产生。按照一定的顺序产生用于兄弟的行，其中，兄弟出现在下面的行组中。

因此，语句 S4 产生表 4 所示的 VEMP 图。VEMP 可用于获得 HierXMLAggregate 运算符中的等级运算对象的值。此外，语句 S4 在“CREATEVIEW VEMP AS”行之后的部分可被用于子查询中，以生成 HierXMLAggregate 运算符中的等级运算对象的值。

硬件概述

图 6 是描述可应用本发明的实施例的计算机系统 600 的框图。计算机系统 600 包括用于传递信息的总线 602 或其它通信装置、用于处理信息的与总线 602 连接的处理器 604。计算机系统 600 还包括主存储器 606，例如随机存取存储器 (RAM) 或者其它动态存储装置，与总线 602 连接，用于储存信息及处理器 604 要执行的指令。

主存储器 606 还可用于储存处理器 604 执行指令过程中的临时变量或其他中间信息。计算机系统 600 还包括只读存储器 (ROM) 608 或者其他静态存储装置，与总线 602 连接，用于储存静态信息和处理器 604 的要执行的指令。存储装置 610，如磁盘或光盘，和总线 602 连接以储存信息和指令。

计算机系统 600 可以经由总线 602 连接到显示器 612，如阴极射线管 (CRT)，用于向计算机用户显示信息。包括字母数字键和其他键的输入装置 614 与总线 602 相连，用于传递通信信息和命令选择到处理器 604。另一种用户输入装置是光标控制 616，如鼠标、跟踪球、或光标方向键，用于传递方向信息和命令选择到处理器 604 及用于控制显示器 612 上的光标移动。这个输入装置通常在两个轴上（第一个轴（例如 X 轴）和第二个轴（例如 Y 轴））具有两个自由度，使装置能指定平面上的位置。

本发明涉及用于实现在此描述的技术的计算机系统 600 的使用。根据本发明的实施例，响应于处理器 604 执行在主存储器 606 中包括的一个或多个指令的一个或多个序列，这些技术通过计算机系统 600 执行。这样的指令可从诸如存储设备 610 的另一计算机可读介质读入主存储器 606。通过执行包含在主存储器 606 中的指令序列，使处理器 604 执行此处所述的处理步骤。在可选实施例中，硬连线电路 (hard-wired circuitry) 可取代软件指令或者与软件指令结合来实施该发明。因此，本发明中的实施例将不限于硬件电路和软件的任何特定组合。

这里使用的术语“计算机可读介质”是指参与提供指令给用于执行的处理器 604 的任何介质。这种介质可以采取很多形式，包括但不限于非易失性介质、易失性介质和传递介质。非易失性介质举例来说包括光盘或磁盘，如存储装置 610。易失性介质包括动态存储器，如主存储器 606。传输介质包括同轴电缆、铜线、和光纤，

包括由总线 602 组成的导线。传输介质还可采取声波或光波的形式，例如那些在无线电波和红外线数据通信过程中产生的声波和光波。

通常的计算机可读介质举例来说包括软盘、柔性盘、硬盘、磁带，或者任何其它磁性介质、CD-ROM、任何其它光介质、打孔纸、纸带、或者任何带孔的物理介质、RAM、PROM、EPROM、FLASH-EPROM、或者其他任何存储芯片或者磁带，或者以下提到的载波、或者计算机可读的任何其他介质。

各种形式的计算机可读介质可参与传送一个或者多个指令的一个或多个序列到用于执行的处理器 604。例如，指令开始可承载远程计算机的磁盘中。远程计算机能将该指令加载到其动态存储器中，然后使用调制解调器基于电话线发送信息。计算机系统 600 本地的调制解调器可接收电话线上的数据，然后使用红外转换器将数据转换成红外信号。红外探测器可以接收红外信号携带的数据，合适的电路可以把信息放到总线 602 上。总线 602 把数据传递到主存储器 606 中，处理器 604 从主存储器 606 取回并执行这些指令。在处理器 604 执行这些指令之前或之后，主存储器 606 接收的指令可随意地储存于存储装置 610 中。

计算机系统 600 还包括连接到总线 602 的通信接口 618。通信接口 618 提供双向数据通信，连接到与局域网 622 相连的网络链路 620。例如，通信接口 618 可以是综合业务数字网 (Integrated Services Digital Network, ISDN) 卡或者调制解调器，用于提供到相应类型电话线的数据通信连接。又如，通信接口 618 可以是局域网 (Local Area Network, LAN) 卡，用于提供至兼容局域网 (LAN) 的数据通信连接。也可以使用无线链路。无论采用何种连接，通信接口 618 均发送和接受承载各种信息的数字数据流的电信号、电磁信号和光学信号。

网络链路 **620** 通常可通过一个或者多个网络提供数据通信给其它数据装置。例如，网络链路 **620** 可通过局域网 **622** 与主机 **624** 连接，或者与互联网服务提供商（Internet Service Provider, ISP）**626** 操作的数据设备连接。ISP **626** 又通过目前通称为“互联网”**628** 的全球分组数据通信网络（World Wide Packet Data Communication Network）提供数据通信服务。局域网 **622** 和互联网 **628** 都使用承载数字数据流的电信号、电磁信号或光学信号。如通过各种网络的信号，网络链路 **620** 上的信号，通过通信接口 **618** 的信号是传输信息的载波的示范形式，这些信号都传送数字数据给计算机系统 **600** 或者传送来自计算机系统的数字数据。

计算机系统 **600** 能通过网络、网络链路 **620** 和通信接口 **618** 发送消息和接收数据（包括程序代码）。例如，在互联网的例子中，服务器 **630** 可通过互联网 **628**、ISP **626**、局域网 **622**、和通信接口 **618**，传送所请求的用于应用程序的代码。

当代码被接收和/或储存在存储装置 **610** 上或者其它非易失性存储器上用于随后执行时，处理器 **604** 可执行所接收到的代码。按照这种方式，计算机系统 **600** 可以获得载波形式的应用程序代码。

以上所述仅为本发明的优选实施例而已，并不用于限制本发明，对于本领域的技术人员来说，本发明可以有各种更改和变化。凡在本发明的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

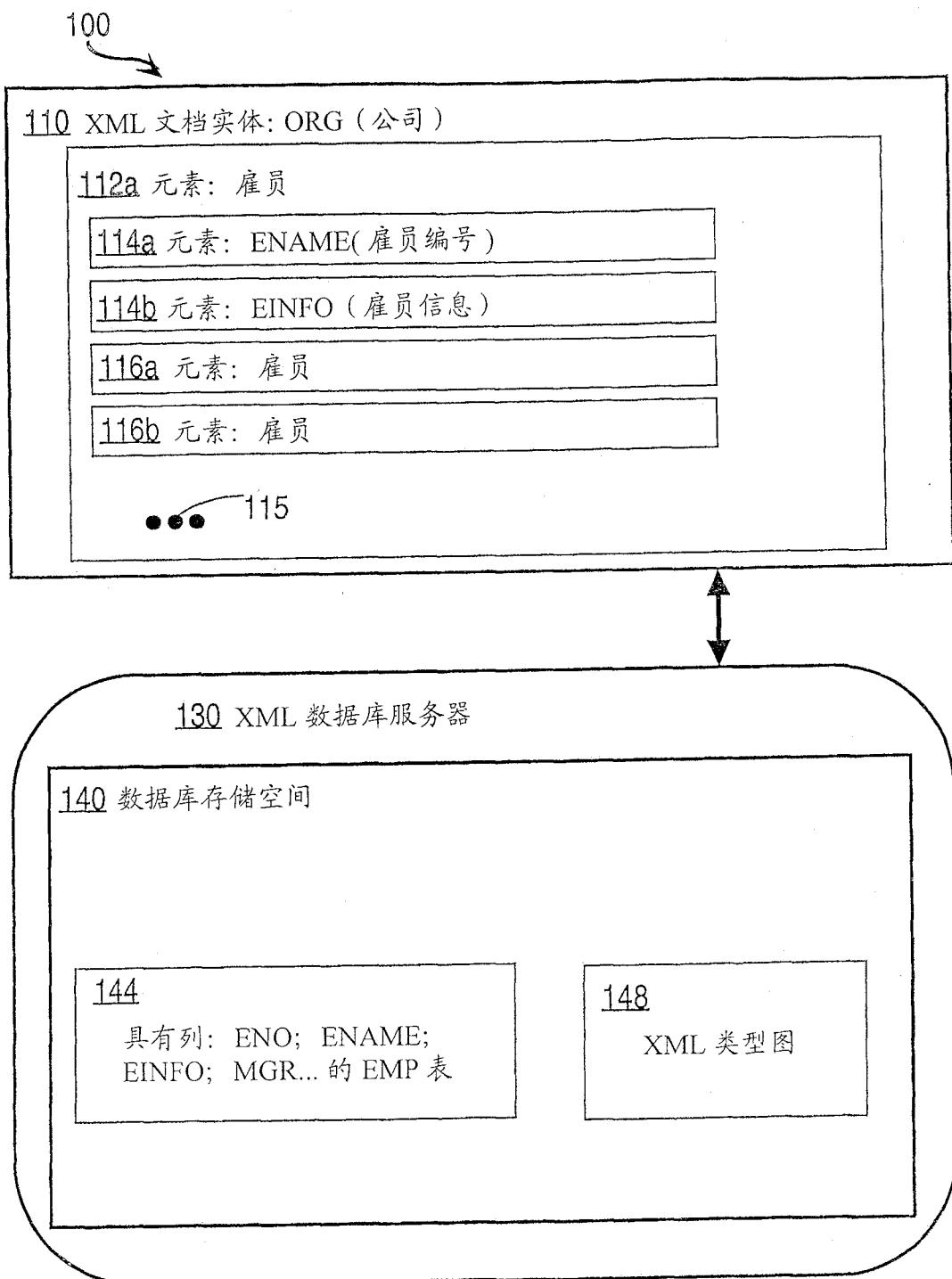


图 1

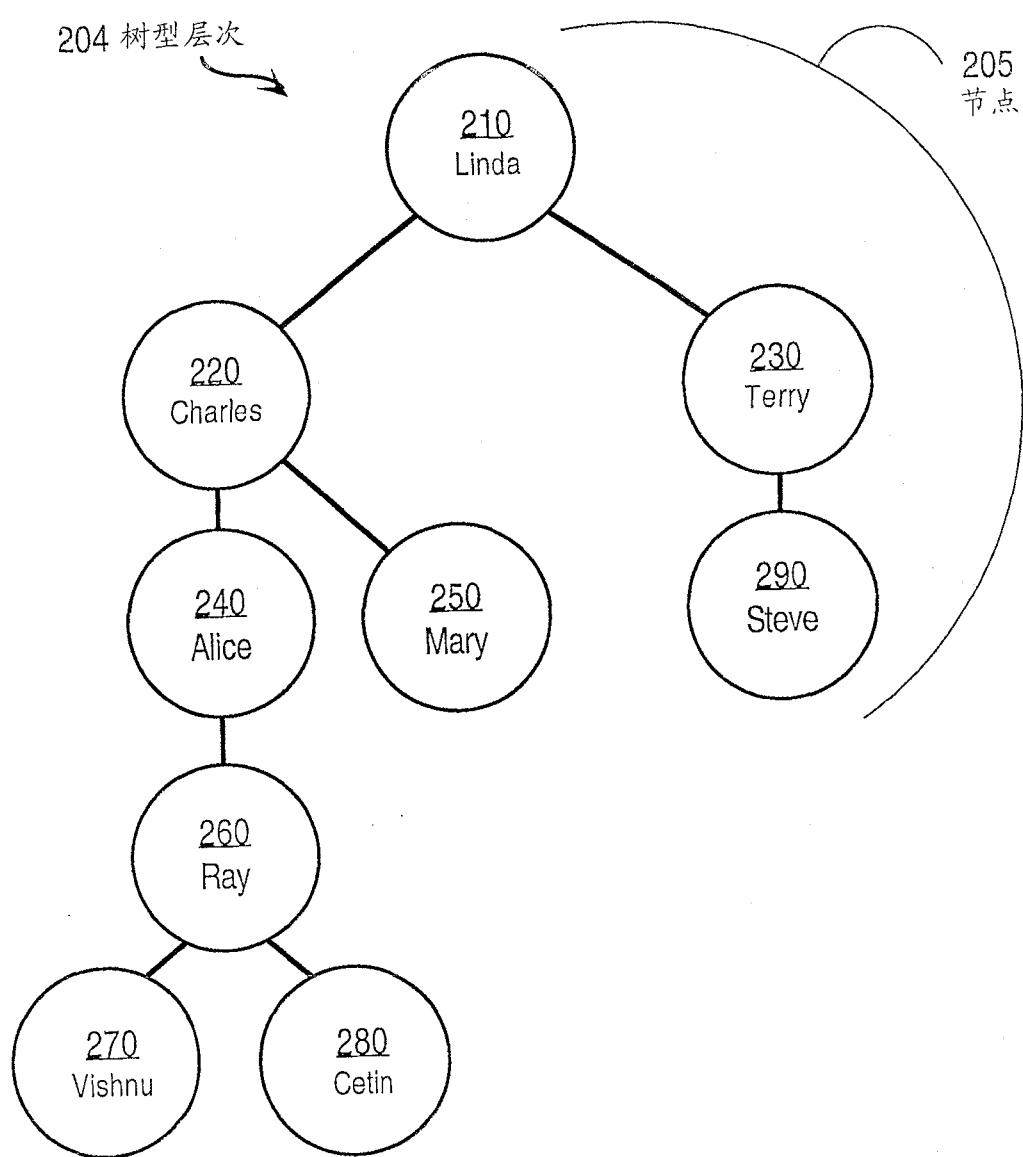


图 2A

202 行数

1	...	201
2	<ORG>	
3	<EMPLOYEE ENO = "1">	
4	<ENAME> Linda </ENAME> <EINFO> ... </EINFO>	
5	<EMPLOYEE ENO = "2">	
6	<ENAME> Charles </ENAME> <EINFO> ... </EINFO>	
7	<EMPLOYEE ENO = "4">	
8	<ENAME> Alice </ENAME> <EINFO> ... </EINFO>	
9	<EMPLOYEE ENO = "6">	
10	<ENAME> Ray </ENAME> <EINFO> ... </EINFO>	
11	<EMPLOYEE ENO = "7">	
12	<ENAME> Vishnu </ENAME> <EINFO> ... </EINFO>	
13	</EMPLOYEE>	
14	<EMPLOYEE ENO = "8">	
15	<ENAME> Cetin </ENAME> <EINFO> ... </EINFO>	
16	</EMPLOYEE>	
17	</EMPLOYEE>	
18	<EMPLOYEE ENO = "5">	
19	<ENAME> Mary </ENAME> <EINFO> ... </EINFO>	
20	</EMPLOYEE>	
21	</EMPLOYEE>	
22	<EMPLOYEE ENO = "3">	
23	<ENAME> Terry </ENAME> <EINFO> ... </EINFO>	
24	<EMPLOYEE ENO = "9">	
25	<ENAME> Steve </ENAME> <EINFO> ... </EINFO>	
26	</EMPLOYEE>	
27	</EMPLOYEE>	
28	</EMPLOYEE>	
29	</EMPLOYEE>	
30	</ORG>	

图 2B

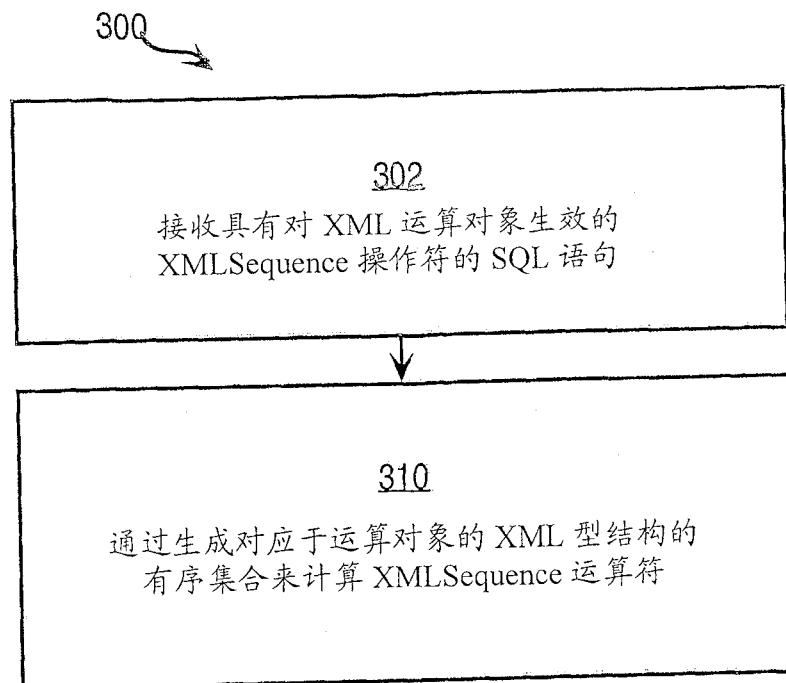


图 3

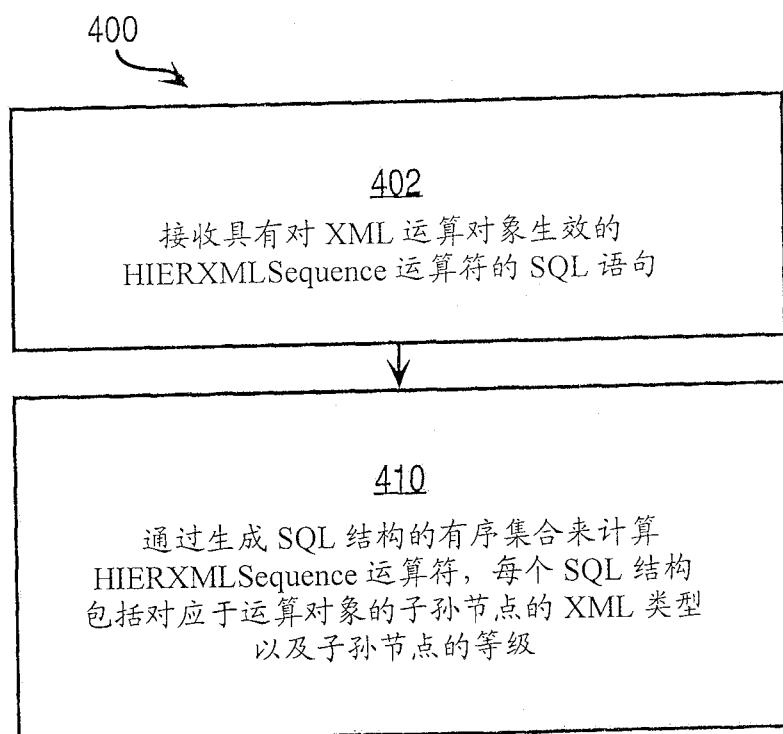


图 4A

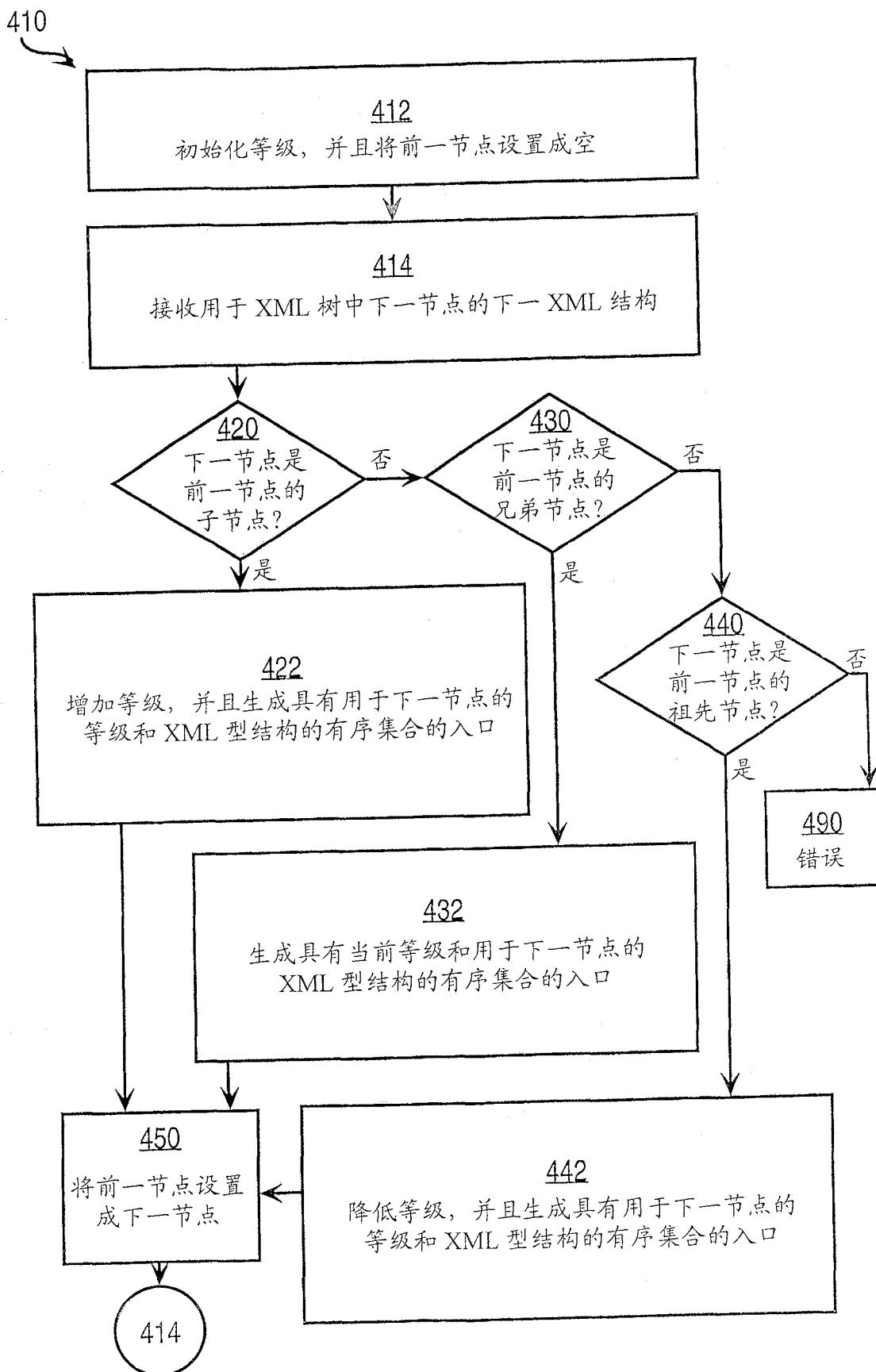


图 4B

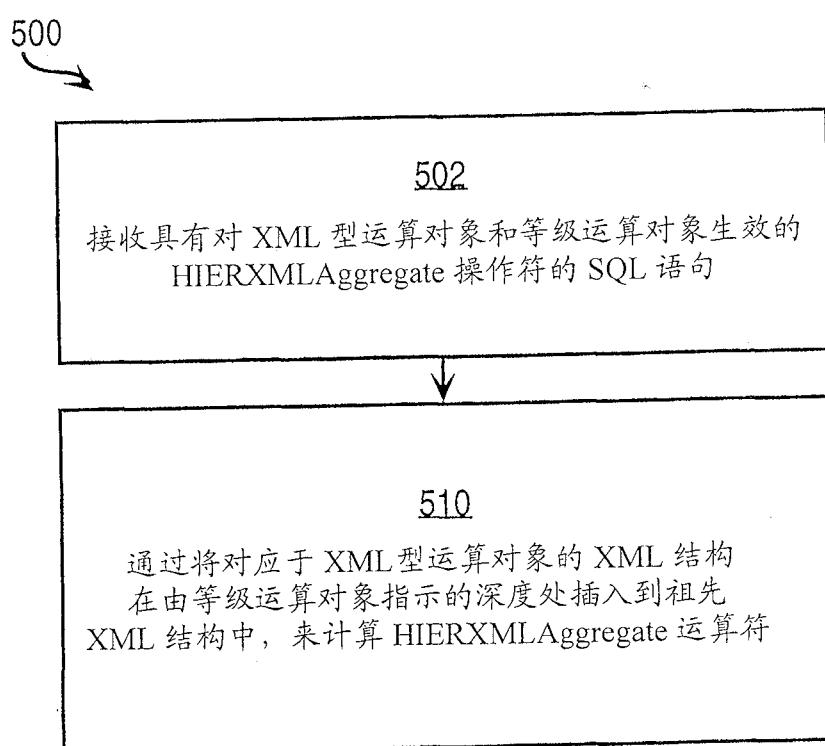


图 5A

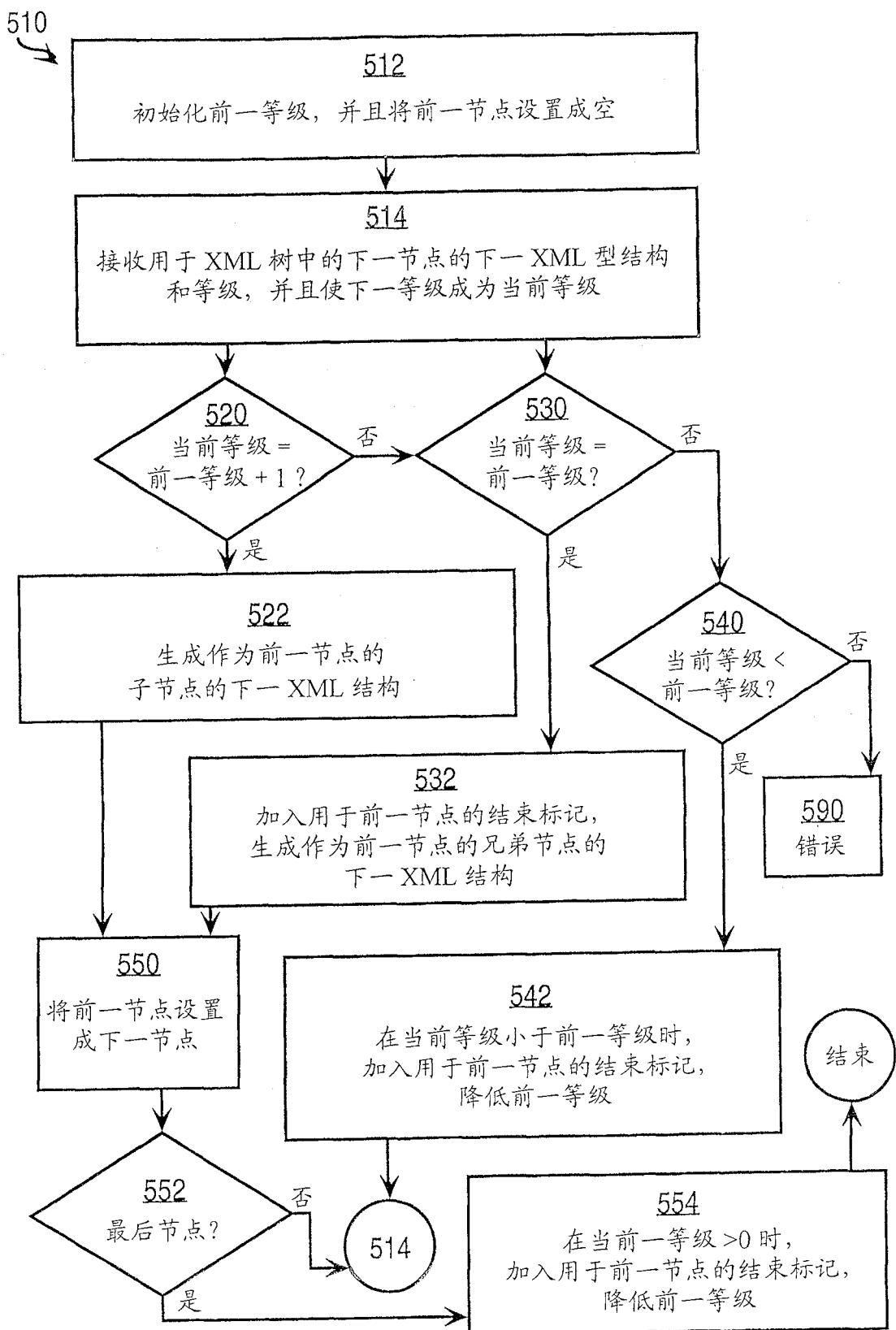
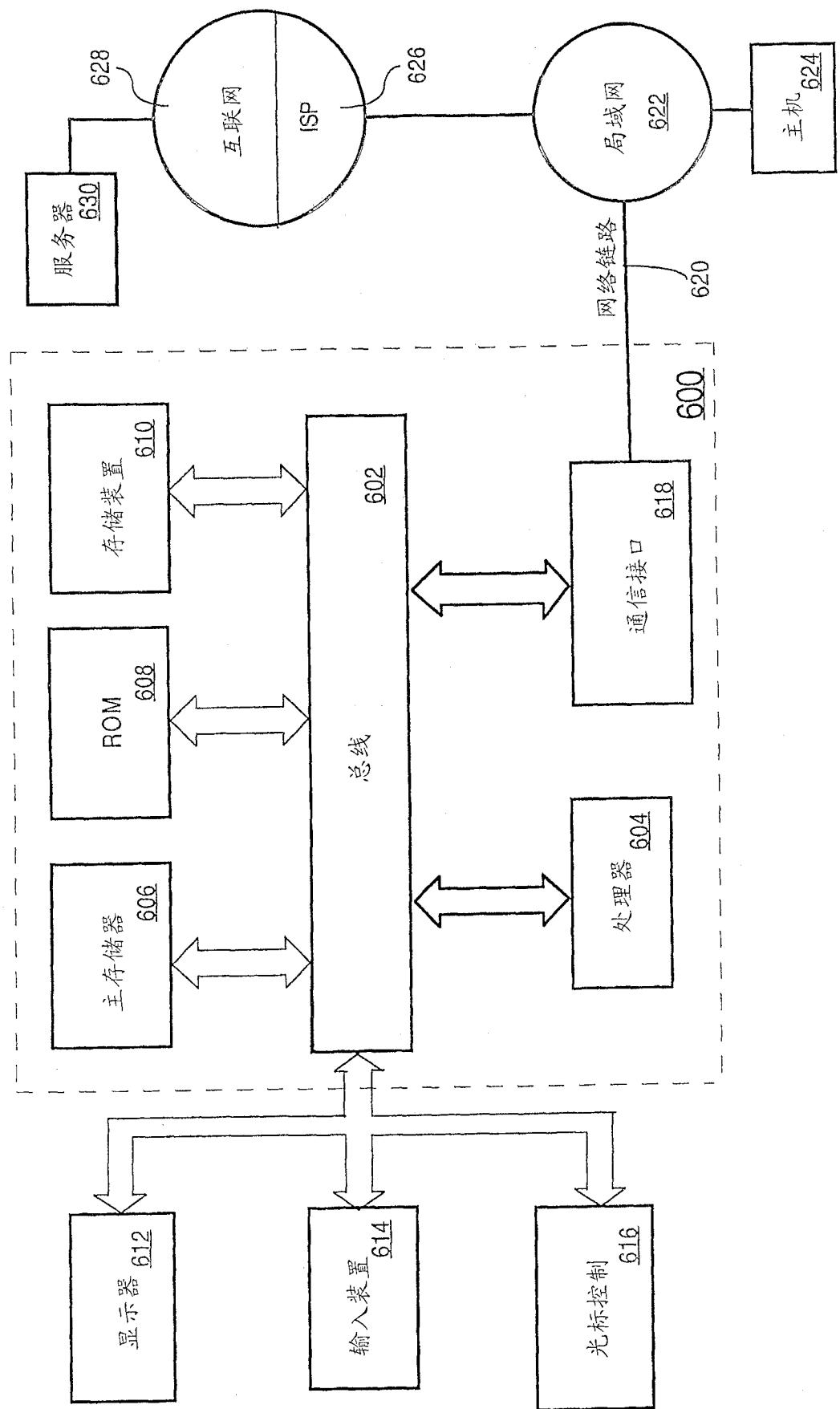


图 5B



6