第 34 卷 第 5 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.5 Computer Engineering March 2008
·软件技术与数据库· 文章编号：1000—3428(2008)05—0077—02 文献标识码：A 中图分类号：TP311

基于关系数据库的 XBRL 存储系统

李如豹[1]，李 军[2]，廖华明[1]
(1. 中国科学院计算技术研究所网格与服务计算研究中心，北京 100086；2. 华北水利水电学院科技处，郑州 450011)

摘 要：可扩展商业报告语言(XBRL)是一种专门用于金融财政系统之间进行数据交换的 XML 扩展语言。该文介绍了一种基于关系数据库的 XBRL 存储系统：X-SIR 系统。该系统可以根据 XBRL 数据模型自动生成关系数据模式，把 XBRL 实例文档中的数据自动转换成数据库表中的数据记录。

关键词：可扩展商业报告语言；XML 语言；关系数据库

XBRL Storage System Based on Relational Databases

LI Ru-bao[1], LI Jun[2], LIAO Hua-ming[1]
(1. Research Centre for Grid and Service Computing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100086;
2. Department of Science and Technology, North China University of Water Conservancy and Electric Power, Zhengzhou 450011)

【Abstract】 eXtensible Business Report Language(XBRL) is a special version of XML for communicating business and financial information. This paper introduces X-SIR, a system for storing and managing XBRL data using a relational DBMS. X-SIR can automatically generate relational schema according to the XBRL data model and automatically transform data in XBRL instance documents into tuples in relational tables.

【Key words】 eXtensible Business Report Language(XBRL); XML; relational database

1 概述

XBRL(eXtensible Business Report Language)是一种面向金融财政领域的XML扩展语言[1]。如何有效存储并使用大量XBRL数据是业务机构需要解决的问题。虽然目前存在各种利用关系数据库系统来存储和查询XML数据的解决方案[2-3]，但XBRL同标准XML的区别使这些解决方案不能直接应用于管理XBRL数据。XBRL语言的数据定义方式和实例文档结构同标准的XML语言不同。XBRL允许开发者定义一组相关的数据项(item)，由XBRL实例文档提供所定义的数据项的具体取值。XBRL使用DTD或XML Schema来定义数据元素和数据项，并采用XLink来描述数据项之间的关系。在XBRL中，数据项的定义包含 2 部分：(1)XBRL schema，定义每个数据项的标识符、名称、类型等信息；(2)XBRL Linkbase，定义多个数据项之间的关系，包括数据项之间的层次、计算、引用关系等。XBRL实例文档虽然是一个标准的XML文档，但数据项并没有组织成复杂的树型结构，而是按特定顺序排列的列表。这使XBRL实例文档的结构呈扁平状，类似在XML文档中只提供所有的叶子节点元素而忽略路径信息。

利用 DBMS 存储 XML 数据或 XBRL 数据须解决 2 个问题：(1)如何利用 XML DTD 或 Schema 来生成关系模式；(2)如何把 XML 数据转换成关系数据库表中的数据记录。存储 XBRL 数据时，还必须考虑 XBRL 的独特数据定义方式和实例文档结构。本文介绍了 X-SIR 系统——基于关系数据库的 XBRL 数据管理系统。

2 X-SIR 的体系结构

X-SIR 包含 2 组核心部件：(1)元数据管理框架。包括 XBRL 元数据解析器、元数据存储管理器和关系模式生成器，其作用是根据输入的 XBRL Taxonomy 和 Linkbase 自动生成关系模式并在目标数据库中创建相应的表和约束信息。(2)XBRL 数据转换引擎。其作用是把输入的 XBRL 实例文档中的数据项转换成关系数据记录并插入到相应的表中。图 1 为 X-SIR 的体系结构。

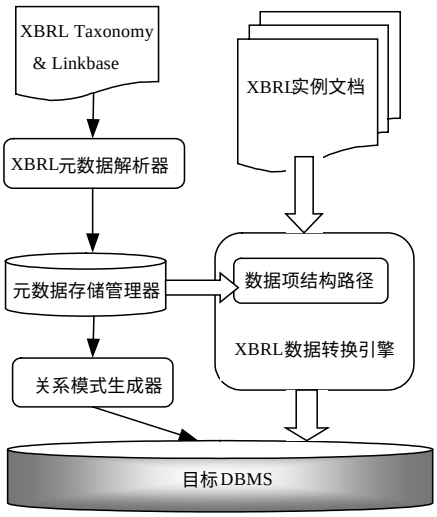

图 1 X-SIR 的体系结构

X-SIR 工作模式分为相应的 2 个部分，分别对应元数据处理和实例文档处理。元数据处理分为 3 个步骤：(1)XBRL 元数据解析器解析所输入的 XBRL Taxonomy 和 Linkbase，从中提取数据项定义；(2)元数据存储管理器利用一套中间的标准格式来记录数据项定义；(3)关系模式生成器通过分析标准

作者简介：李如豹(1979－)，男，博士研究生，主研方向：信息网格，信息集成；李 军，讲师；廖华明，副研究员
收稿日期：2007-05-21 **E-mail：**lirubao@software.ict.ac.cn

格式的数据项定义生成关系模式，并在目标 DBMS 中创建表和约束。实例文档处理由转换引擎完成。转换引擎需要使用元数据存储管理器记录的数据项定义信息来决定实例文档中的每个数据项的上下文结构，从而决定如何在目标数据库中记录每个数据项携带的数据。

在 X-SIR 中，元数据存储管理器的作用类似编译器中的中间代码。XBRL 利用 Taxonomy 和 Linkbase 文档定义多个数据元素及元素间层次关系。由于 XBRL 自身规范的发展演化以及具体业务系统的升级变更等原因，不同版本的 Taxonomy 和 Linkbase 文档的结构和数据定义方式可能发生变化，因此 X-SIR 利用一个中间元数据存储管理器来屏蔽具体数据定义方式的变化。元数据存储管理器定义了一套标准的数据项表示格式和存储方式，而不同的前端 XBRL 元数据解析器可以被替换以处理不同版本的 Taxonomy 和 Linkbase 文档。

3 关系模式的自动生成

在 XBRL 中，多个数据项之间的层次关系构成 1 棵树，其中，每个节点表示一个数据项；每个边表示 2 个数据项之间的父子关系。关系模式生成器只须考虑如何根据这一层次树来生成关系模式。由于在 XBRL 针对的金融信息领域中，每个数据项都有一个全局唯一的名称，因此层次关系不会形成一个有向无环图。例如虽然董事会秘书和证券事务代表两者均有“姓名”这一数据项，但两者使用不同的数据项名称，分别为 DongShiHui MiShuXingMing 和 ZhengQuanShiWuDai Biao XingMing。

树中的每个边可能是一对一关系，即父节点只能包含一个相应的子节点；也可能是一对多关系，即父节点可能包含多个相应的子节点。这决定了生成的关系模式的不同。X-SIR 采取一个类似Hybrid Inlining[2]的技术来生成关系模式。Hybrid Inlining的基本思路如下：对于树中的每个节点，如果其父节点同该节点间存在一对多的关系，那么为该节点创建一个单独的表，并在该表同其父节点对应的表之间创建一个主外键约束；否则，使该节点成为其父节点对应的表中一个列，即内联(inlining)。由于其父节点可能不是对应一个表而是被内联成一个列，因此上述过程应该寻找该结点对应的最近祖先节点的表。

因为 XBRL 中每个数据项都对应一个业务实体，所以要为每个数据项创建单独的表，使该数据项对应的数据能被独立地存储并查询。这样做将导致目标数据库中存在大量的表和主外键约束。因此，X-SIR 采取折中的设计思路，把 Hybrid Inling 技术修改成：只有叶子节点可以被内联。即对于树中的非叶子节点，不管其父节点与它之间是一对一关系还是一对多关系，均为其创建单独的表并创建相应的主外键约束。图 2 展示了一个数据项层次结构树。

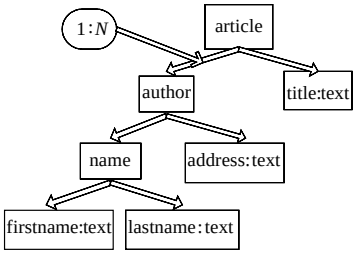

图 2 示例数据项结构

根据 X-SIR 修改的 Hybrid Inling 技术，创建如下 3 个表：

(1)article(parent_id: int, self_id: int, title: text);

(2)author(parent_in: int, self_id: int, adderss: text);

(3)name(parent_id: int, self_id: int, firstname: text, lastname: text)。

每个表均存在 2 个预置的列：parent_id 和 self_id。其中，self_id 是每条记录的标识符，并且是该表的主键；parent_id 同该表的父表中的 self_id 构成主外键约束，这一约束保证了数据记录的完整性。对于每个数据项，存在唯一的结构路径，即从数据项结构树的根节点到该数据项对应的节点的路径。例如数据项 firstname 的结构路径是：/article/author/name/firstname。元数据存储管理器缓存了每个数据项的结构路径信息，供实例文档转换引擎使用。

4 XBRL 数据的转换

数据转换引擎负责把输入的 XBRL 实例文档中的数据项转换成关系数据记录并插入到相应的表中。XBRL 实例文档也是 XML 文件，但它具有扁平状结构，即所有数据项在同一层次被依次列出，而实例文档不提供数据项的结构信息。转换引擎把每份 XBRL 实例文档看作一个数据项序列，并流式地逐一处理每个数据项。

对每个数据项，转换引擎需要从元数据存储管理器中读取该数据项对应的结构路径，从而决定该数据项在数据库表中的位置。例如，当转换引擎读取当前数据项 X 时，转换引擎必须获取数据项 X 的结构路径：/A/B/C/X。这一路径表明，数据项 X 是表 C 的一个字段，而表 C、表 B、表 A 则依次存在主外键约束。为了存储数据项 X，转换引擎首先要找到表 C 中相应的记录，然后在对应 X 列的单元格中存储数据项。如果不存在相应的记录，则转换引擎要在表 C 中插入 1 条新的记录并存储 X。由于主外键约束的存在，使得对表 C 的插入操作必须考虑其父表 B 中是否存在相应的记录，如果不存在则首先要创建。该过程依次递归，直至表 A。

由于 XBRL 应用在金融财政领域，因此 XBRL 实例文档中的每个数据项都有一个特殊的上下文信息。一个 XBRL 实例文档片断如下：

<cnfr-pt: YingFuGongZi contextRef=C_instant_20041231>37280406.08</cnfr-pt: YingFuGongZi>

<cnfr-pt: YingFuGongZi contextRef=C_instant_20041231_mu>33029117.04</cnfr-pt: YingFuGongZi>

可以看出，每个数据项在实例文档中是一个 XML 元素，其子节点是该数据项的取值，元素名称是该数据项的表示符。每个元素有一个属性“contextRef”，表示该数据项取值的上下文信息，用于解释该数据项。在上述片断中，contextRef“C_instant_20041231”表示该数据项(应付工资)取值有效时间是 2004 年 12 月 31 日；而 contextRef“C_instant_20041231_mu”除提供该时间信息外还表示该数据项针对的是母公司的数据。由于数据项的取值只能通过其 contextRef 来解释，因此存储每个数据项时必须考虑 contextRef 的一致性。

为了正确存储数据项上下文信息，X-SIR 采用了 2 个原则：(1)每个表中同一条记录中的每个字段必须有相同的上下文信息；(2)如果相关表的 2 条记录间存在主外键约束，那么这 2 条记录必须具有相同的上下文信息。采取这 2 个原则可以保证数据项上下文的完整性和一致性，避免了转换后的数据记录将丢失上下文信息而无法被解释。

（下转第 81 页）