

# XML 查询的推理审计

严和平 刘 兵 汪 卫 施伯乐

(复旦大学计算机与信息技术系 上海 200433)

**摘 要** XML 文档作为一种网上信息交换方式,其应用越来越广泛. 信息发布的安全性给数据库带来新的挑战,目前一些安全策略以法律条文形式颁布,这要求采用有效的手段证实对 XML 文档的访问与安全策略的一致性. 审计能达到这样的目的,但已有的审计方法只能对 SQL 查询结果进行审计,不能对 XML 文档查询——XQuery 或 Xpath 进行审计,且蓄意破坏的用户可能通过对查询结果进行推理来访问敏感信息,这就要求对 XQuery 的审计必然同时具备推理能力. 对此,首先提出了可靠而可行的 XQuery 审计方法、算法及相应查询图模型(QGM);为使审计具备基本的推理能力,针对 XML 文档的几种典型约束,给出了推理审计方法、算法及相应查询图模型;实验结果表明,给出的 XML 查询推理审计框架切实可行.

**关键词** XML; XQuery; 查询; 审计; 推理  
中图法分类号 TP311

## Inference Auditing the XQuery of XML

YAN He-Ping LIU Bing WANG Wei SHI Bo-Le

(Department of Computer and Information Technology, Fudan University, Shanghai 200433)

**Abstract** XML(eXtensible Markup Language) is rapidly becoming the de facto standard for exchanging data between applications, and publishing data on the Web brings security database new challenges. Privacy principles are even being mandated internationally through legislations and guidelines, and this requires the secure database to verify that it adheres to its declared data disclosure policy. Auditing system satisfies the above desiderata well, but existed auditing system can only be used for SQL query and not fit for the XQuery or Xpath of XML. Moreover only auditing the result of XQuery is not enough, because malicious user can access sensitive information by inferring the result of XQuery. This demands the auditing system have the basic inference capacity. Firstly based on the existed auditing system, the authors propose their XQuery auditing system, and then they add the inference capacity to the audit framework. Their experiment results show the effectiveness and efficiency of the proposed XQuery audit method, algorithm, and the corresponding Query-Graph-Model.

**Keywords** XML; XQuery; query; audit; inference

## 1 引 言

作为一种网上信息发布的方式,XML 文档已迅

速成为许多应用之间数据交换的一种标准. 在与合作方信息共享的同时,如何保护各自的敏感信息,就成了普遍关注的问题. 关于 XML 文档的访问控制,仍是科研工作者们研究的热门话题<sup>[1~3]</sup>. 在保证信

收稿日期:2006-04-04;修改稿收到日期:2006-06-01. 本课题得到国家自然科学基金(60303008,69933010)、国家“八六三”高新技术研究发展计划项目基金(2002AAA423430)资助. 严和平,男,1970年生,博士,主要研究方向为安全数据库、XML、数据挖掘. E-mail: 031021055@fudan.edu.cn; hpyan\_bj@163.com. 刘 兵,男,1978年生,博士,主要研究方向为数据挖掘、数据仓库、数据库理论与应用、可信系统与网络. 汪 卫,男,1970年生,教授,博士生导师,主要研究领域为数据库、安全数据库、数据挖掘、XML 数据库. 施伯乐,男,1936年生,教授,博士生导师,主要研究领域为数据库与知识库、数据挖掘、数字图书馆、安全数据库.

息安全的同时,必须保证信息的最大可用性,这种细粒度访问控制的要求<sup>[4]</sup>与防止敏感信息的泄露就形成了一对矛盾.而且,对发布的信息,如何证明用户对敏感信息的访问是遵守了相应保密准则,也是所要解决的问题.对访问 XML 文档的各种 XQuery 进行审计能验证保密准则的有效性,且这种事后追究责任的审计不仅可以威慑内部人员的非法访问(内部人员利用职权犯罪),还可以检查是否存在恶意的木马攻击,从而提高信息的安全性,但常规的审计方法<sup>[5]</sup>不一定适于 XML 文档查询的审计.并且,如果只是简单地对查询结果进行审计,会有许多不足之处,因为蓄意破坏的用户可能通过访问 XML 文档中他所能访问的信息,或者多个用户之间相互交换查询结果信息,然后利用各种 XML 文档约束进行推理从而达到访问敏感信息的目的.因此,首先必须设计能处理 XML 查询的审计模型,然后对其审计能力进行提升,使之具有初步的推理审计能力.

### 1.1 相关工作及其局限性

关于 XML 文档信息发布的安全性,已有大量的研究成果<sup>[2,3]</sup>.这些成果主要是从加密的技术角度来考虑敏感信息的安全问题,但这些加密措施在保护敏感信息的同时,也一定程度地降低了信息的可用性,而且访问控制的粒度越细,实现起来就越困难,查询访问的效率就会越低.而由于 XML 文档自身方面的约束所产生的推理通道仍然可能导致敏感信息泄露. Yang<sup>[1]</sup>对颠覆 XML 文档上敏感信息的推理通道进行了专门的研究,针对三种典型的约束所产生的推理通道提出了相应的防范措施,但恶意的用户可能利用各种历史查询信息进行推理,或者多个非法用户相互之间交换查询信息进行推理.

审计是一种事后行为,强有力的审计措施可允许适当放宽信息的访问要求,将查询阶段的部分保密措施放在审计阶段来实现,有效提高信息的可用性. Hippocratic<sup>[6]</sup>给出了一个负责管理保密信息的数据库系统构想. Agrawal<sup>[5]</sup>针对该数据库提出的十条保密准则中的第十条:要求数据库能够证明它遵守了信息发布的保密准则,给出了相应的审计体系结构.该审计方法主要审计以往关系数据库中的查询是否访问特定的数据.审计直接加在常规的查询处理之上,开销小,能快速而准确定位访问了特定数据的查询.审计的粒度可以是记录的单个属性.审计的语言使用 SQL 语言形式,直观且易于使用.但该审计方法是针对一般关系数据库的,而 XML 文

档是一种半结构化信息载体,不同于一般数据库,其存储、查询方式不同,而且导致敏感信息泄露的各种推理通道可源自不同的 XML 文档约束.

因此,有必要将常规的审计方法拓展到 XML 查询,不仅对 XML 查询结果直接进行审计,而且要对利用各种文档约束推理查询结果的推理信息进行审计,使一般的审计方法具有基本的推理能力.

### 1.2 本文的贡献

本文以 Agrawal<sup>[5]</sup>所给出的审计方法为基础,保留原有审计方法的特点:方便、细粒度、快速而精确等,提出 XML 查询的审计框架,并针对原有审计框架中审计能力的不足,结合常见的、易导致敏感信息泄露的几种推理方法,对 XML 查询的审计方法进行了研究,提出了能审计 XML 查询具备推理能力的审计框架,使审计功能更加强大.主要贡献体现在以下几个方面:(1)给出了对 XML 查询进行审计的方法以及相对应的算法和查询图模型(Query Graph Model, QGM). (2)提出了能利用常规的 XML 文档约束如父子约束、祖孙约束和函数依赖进行推理的审计方法、算法和查询图模型(QGM). (3)提出了 Xquery 查询的审计和推理审计体系结构. (4)审计以及推理审计实验证实了其可行性和高效性.

本文第 2 节先给出 XML 文档、XQuery 查询、审计相关的定义与理论,然后对支持审计方法得以实现的 XML 文档备份数据库进行介绍;第 3 节详细介绍对 XML 查询进行审计的审计方法,并结合实例给出相对应的算法和 QGM;第 4 节结合常规的 XML 文档约束,给出了具备推理能力的审计方法,并结合实例给出相对应的算法和 QGM;第 5 节给出具备推理能力的 XML 查询审计框架;第 6 节是 XML 文档查询的审计以及推理审计实验.第 7 节是小结.

## 2 术语的定义及相关理论

XML 迅速成为许多应用网上信息发布之间数据交换的一种标准,越来越得到了广泛的应用,下面给出形式化的定义.

### 2.1 DTD 和 XML 文档的树模型以及与之对应的查询

定义 1. DTD 是一个五元组  $(E, A, M, N, r)$ , 其中:

$E$  是有限的元素类型集合;  $A$  是有限的属性类

型集合,属性和元素不同名;存在一个特殊的属性  $id \in A$ ; 对  $\forall e \in E, M(e)$  称为  $e$  的元素类型定义.  $M(e) = S$  (字符串), 或者是一个正规表达式  $\alpha ::= \epsilon | e' | \alpha, \alpha | \alpha^*$ . 其中  $\epsilon$  表示空,  $e' \in E$ , “,” 和 “\*” 分别代表连接和闭包; 对  $\forall e \in E, N(e)$  称为  $e$  的属性类型定义,  $N(e) \subseteq A$ . 对于  $\forall e \in E, id \in N(e); r \in E$ , 称为根元素类型. 若  $e, e' \in E, e'$  属于  $M(e)$  所定义的字母表, 且在  $M(e)$  中不包含  $e'$  的自连接, 则称  $e'$  相对于  $e$  是唯一的.

**定义 2.** 符合给定  $DTD(E, A, M, N, r)$  的 XML 文档的树模型:

(1)  $V$  表示有限的节点集,  $S$  表示字符串; (2) 对每一个节点  $v$ , 定义函数  $name(v) \in E \cup A$ . 根据  $name(v)$ , 进一步定义两个  $V$  的子集:  $V_e = \{v | v \in V, name(v) \in E\}; V_a = \{v | v \in V, name(v) \in A\}$ ; (3) 对  $V_e$  中的节点, 定义函数  $subelem(v)$  是列表  $[v_1, v_2, \dots, v_n] (v_i \in V_e)$ , 若  $name(v) = e, M(e) = \alpha$ , 则  $name(v_1), name(v_2), \dots, name(v_n)$  属于  $\alpha$  定义的正正规集; (4) 对于  $V_e$  中的节点, 定义函数  $attr(v)$  是集合  $\{v_1, v_2, \dots, v_m\} (v_i \in V_a)$ . 对任意  $a \in N(name(v))$ , 存在唯一的  $i \in [1, m]$ ,  $name(v_i) = a$ ; (5) 对每一个

节点  $v$ , 定义函数  $value(v) \in \{S\}$ . 若  $name(v_1) = name(v_2) = id$ , 则  $value(v_1) \neq value(v_2)$ ; (6) 有且仅有一个特殊的节点, 记为根节点  $root, name(root) = r$ .

XQuery 提供了 7 种表达式, 但一般来说 FLWR 表达式最常用也最被感兴趣. FLWR 是 XQuery 关键字 For-Let-Where-Return 的首字母缩略词. FLWR 表达式相当于 SQL 中的 SELECT...FROM...WHERE... 语句, For 子句中的表达式指定查询中所用到的 XML 文档, Let 子句定义查询过程中所用到的变量, Where 子句定义条件表达式, Return 表达式定义最终查询返回的结果. 下面用一个例子来说明 Xquery 查询的使用方法, 并将其用于后面的审计中.

**例 1.** XML 文档树模型 (mdb.xml)<sup>[7]</sup> 如图 1 所示, 查询“Bette Davis”参加过演出, 并被提名为“Oscar”奖的喜剧电影名称 (name). 查询表达式可表示为  $Q$ :

```
for $m in document("mdb.xml")//movie-genre
  [name="Comedy"]//movie[.//actor/name="
    "Bette Davis"]
where contains($m/movie-award/name, "Oscar")
return <m/name>{$m/name}</m-name>
```

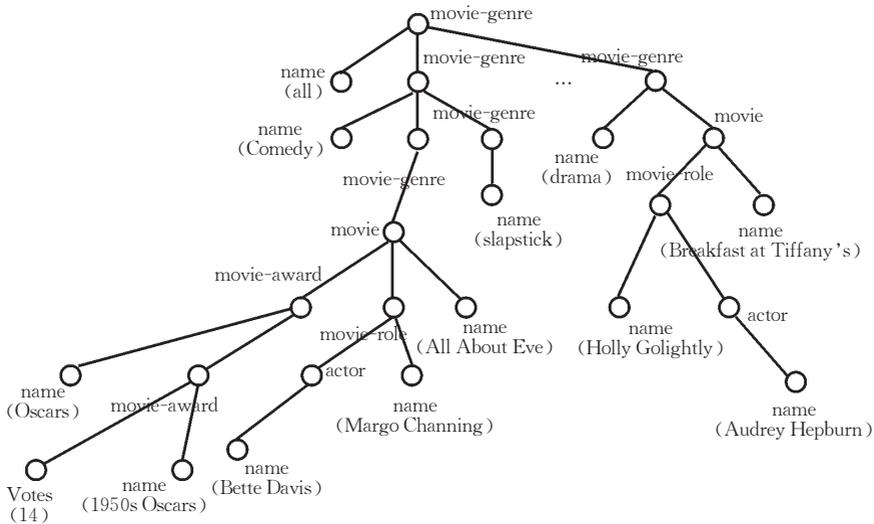


图 1 XML 树文档模型

2.2 审计表达式及与之相关的定义

审计的目的就是要找出访问了特定数据的 XQuery 查询, 这里给出的审计表达式非常接近于 Xquery 查询语句语法, 熟悉 Xquery 语法的审计员很容易就能掌握 XML 查询审计的定义.

审计表达式定义为

```
audit    audit list
for      XML list
where    condition list
```

其中 audit 对应 XQuery 关键词中的 return, 后面跟审计列表, 用于定义要审计的特定数据, 为了能够处理各种不同的文档树模型如 Shallow Tree<sup>[7]</sup>, for 语句后面可以是多个 XML 文档的连接, where 后面紧跟条件表达式列表. 审计最终返回的结果为访问了审计列表中的数据且使审计条件表达式为真的查询.

下面是一个审计的例子.

**例 2.** 审计访问了例 1 中 XML 文档的查询,

若查询访问了审计列表中的数据,且使条件表达式为真,就将该查询作为审计的结果.

```
audit  <m/name>{$m/name}</m-name>
for    $m in document("mdb.xml")//movie-genre
      [name="Comedy"]//movie[.//actor/name=
      "Bette Davis"]
where  contains($m/movie-award/name,"Oscar")
```

显然,例 1 中的查询访问了审计表达式中的数据, $Q$  为最终的审计输出结果.下面给出与审计相关的一些表达式的形式化定义.

为了表示方便,且易于理解,在一些地方, $X_{Query}$  查询  $Q$  的形式化记法仍沿用某些 SQL 的形式化记法, $X_{Query}$  和审计形式化表示如下:

$Q = \mathcal{R}_{ON}(\sigma_{P_Q}(D_1 \times D_2)); A = \mathcal{A}_{ON}(\sigma_{P_A}(D_1 \times D_2))$ , 其中  $D_1, D_2, D_3$  可以是多个 XML 文档之间的连结,如  $D_1 = D_{11} \times D_{12} \times \dots \times D_{1n}$ .  $R_N$  表示出现在  $X_{Query}$  查询中的各种数据集合,包括 for 子句和 where 子句列表中的数据,  $\mathcal{R}_{ON}$  表示出现在查询表达式里 return 子句中的数据集合;用  $\mathcal{A}_{ON}$  表示出现在审计列表中的 XML 文档数据集合.  $P_Q$  表示  $X_{Query}$  查询  $Q$  中的条件谓词,  $P_A$  表示审计表达式中的条件谓词.

**定义 3.** 关键子树( $Cri(T, Q)$ ). XML 文档树中的子树  $T$  为  $X_{Query}$  查询的关键子树,当且仅当略去该子树  $T$  时,将产生不同的查询结果.

**定义 4.** 最小关键子树( $MinCri(T, Q)$ ). 如果关键子树只包含与查询表达式相关的各种数据,那么就称该关键子树为最小关键子树.

有关关键子树和最小关键子树的概念同样适用于审计.

如果  $X_{Query}$  查询  $Q$  访问了审计列表中指定的数据,那么查询  $Q$  就是应该审计的对象.

**定义 5.** 候选查询( $cand(Q, A)$ ). 对审计表达式  $A$  而言,  $X_{Query}$  查询为候选查询,当且仅当  $R_N \supseteq \mathcal{A}_{ON}$ .

并不是所有的候选查询都是审计的结果,只有当候选查询同时也满足审计中的条件表达式时,才会最终成为所要审计的结果.

**定义 6.** 可疑查询( $susp(Q, A)$ ). 如果候选查询和审计有公共的最小关键子树,则  $X_{Query}$  查询  $Q$  为可疑查询,即  $(susp(Q, A)) \Leftrightarrow \exists T \in D, D$  为 XML 文档树模型,使  $(MinCri(T, Q)) \wedge (MinCri(T, A)) \neq \emptyset$ , 则  $Q$  为可疑查询,其中  $D$  可以是多个 XML 文档的连结,即  $D = D_1 \times D_2 \times \dots \times D_n$ .

由于现在安全策略可以定义提交查询的用户能访问的信息和提交查询的目的,完整的审计表达式定义如下:

```
otherthan  purpose-recipient pairs
during     start-time to end-time
audit      audit-list
for        XML document list
where      condition list
```

其中 otherthan 用于审计那些不符合信息发布规范的查询,during 用于指定审计的时间段,表示只对该时间段内提交的查询进行审计.

### 2.3 支持审计的系统结构

审计过程中,审计工作是和审计目标和审计时间密切相关的,这就要求查询日志和 XML 文档数据库的存储能够对这样的审计做出相应支持.

#### 审计日志

审计系统维持着一张关于过去各种查询的日志,记录了提交查询的用户 ID,提交查询的时间以及与之相对应的查询表达式,通过对查询表达式的静态分析,排除那些非候选查询,减少审计的工作量.查询日志以表的形式存放在系统中.

#### 备份 XML 文档数据库

在判断某候选查询是否为可疑查询时,可以对 XML 文档树有选择地进行回放,重建提交查询时刻的 XML 文档树模型状态,这是备份数据库所要完成的目标.

在实现的过程中,对 XML 文档的备份存储结构做一些修改,在主要存储文档中结点信息的同时,给每个结点增加两个属性信息:结点建立的时间 TS 和与之相对应的操作 OP,OP 取 {'insert', 'delete', 'update'} 值,对 XML 文档树结点的各种操作,使用三个触发器来对结点的存储进行更新,插入触发器响应将结点插入到 XML 文档树,执行向存储 XML 文档树的备份数据库( $T^b$ )中插入元组,并置插入结点的 OP 值为 'insert';更新触发器响应对 XML 文档树结点进行修改的操作,并更新  $T^b$ :先插入一个结点,然后将其 OP 值设置为 'update';删除触发器响应从 XML 文档删除结点的操作,并完成从  $T^b$  删除结点信息的动作,在删除结点之前,先插入一个结点,然后将该结点的 OP 值设置为 'delete',所有这三种情况,都将新结点 TS 的值设置为操作执行的时间.为了能够恢复到  $\tau$  时刻  $T^b$  的状态,需要产生  $\tau$  时刻  $T$  的快照,通过在日志备份  $T^b$  上定义视图  $T^\tau$  来实现:

$$T^\tau = \pi_{p, c_1, \dots, c_m} (\{t | t \in T^b \wedge t.TS \leq \tau \wedge t.OP \neq \text{'delete'} \wedge \exists r \in T^b \text{ s.t. } t.p = r.p \wedge r.TS \leq \tau \wedge r.TS > t.TS\}).$$

对不同的键值,  $t.T^\tau$  最多只含  $T^b$  中的一个结点, 对  $T^b$  中有相同键值的一组结点, 选中的结点  $t$  是在  $\tau$  时刻或之前创建的, 且不是已删除的结点, 这其中没有其它有相同主键的结点  $r$ , 结点  $r$  是在  $\tau$  时刻或在此之前创建的, 但其创建时间却迟于  $t$  的创建时间. 根据  $\tau$  时刻视图  $T^\tau$  即可以建立  $\tau$  时刻提交查询时的 XML 文档树状态了. 对具体存储结点信息的备份日志数据库, 其索引采用时标组织形式 (time stamped organization) 和延迟更新 (Lazy) 的方法<sup>[5]</sup>, 延迟更新策略是只在审计时才更新索引, 否则  $T^b$  处于一种无序状态.

### 3 XML 查询的审计

#### 3.1 审计步骤

对 XML 查询进行审计主要包括以下几步:

1. 从查询日志中选择候选查询, 即  $R_N \supseteq A_{ON}$  的查询, 而将其它查询排除.
2. 审计查询的时间戳, 将不在审计表达式 during 子句时间间隔内的查询排除.
3. 审计查询的 purpose-recipient, 看是否和审计表达式中的 purpose-recipient 相匹配.
4. 产生审计查询结果, 将审计表达式的条件  $P_A$  用于

第 3 步所得到的查询结果 (候选查询), 得到审计查询 (可疑查询), 将所有得到的审计查询结果合并输出.

#### 3.2 审计方法

对单个 XML 文档的查询和审计 (图 2 是一个具体过程).

**定理 1.** 假设  $D$  为 XML 文档树数据模型,  $A = R_{ON}(\sigma_{P_A}(D))$  为审计表达式,  $Q = A_{ON}(\sigma_{P_Q}(D))$  为候选查询, 当且仅当  $(\sigma_{P_A}(\sigma_{P_Q}(D))) \neq \emptyset$  时,  $Q$  为可疑查询.

定理的证明可以采取 Agrawal<sup>[5]</sup> 文中引理 1 的证明方法, 二者的区别在于本文定理中的查询为对 XML 文档的 XQuery 查询  $Q$ , 审计是针对 XQuery 查询来进行审计, 在此略去其证明过程.

在实际对 Xquery 查询  $Q$  进行审计的过程中, 由于组织 XML 文档信息的方法有多种, 在对文档信息进行查询时, 可能涉及到多个 XML 文档的连接操作, 如以 Shallow-1<sup>[7]</sup> 方式来组织 XML 文档.

**定理 2.**  $D_1, D_2, D_3$  分别为 XML 文档树模型,  $Q = R_{ON}(\sigma_{P_Q}(D_1 \times D_2))$  为候选查询, 对审计表达式  $A = A_{ON}(\sigma_{P_A}(D_1 \times D_3))$  而言, 该 XQuery 查询  $Q$  为可疑查询, 当且仅当  $(\sigma_{P_A}(\sigma_{P_Q}(D_1 \times D_2 \times D_3))) \neq \emptyset$  时.

**证明.** 按照可疑查询的定义,

$susp(Q, A) \Leftrightarrow \exists$  最小关键子树  $d_1, d_1 \in D_1, d_1$  同时为  $Q$  和  $A$  的最小关键子树, 即有  $MinCri(d_1, A) \wedge MinCri(d_1, Q)$ ;

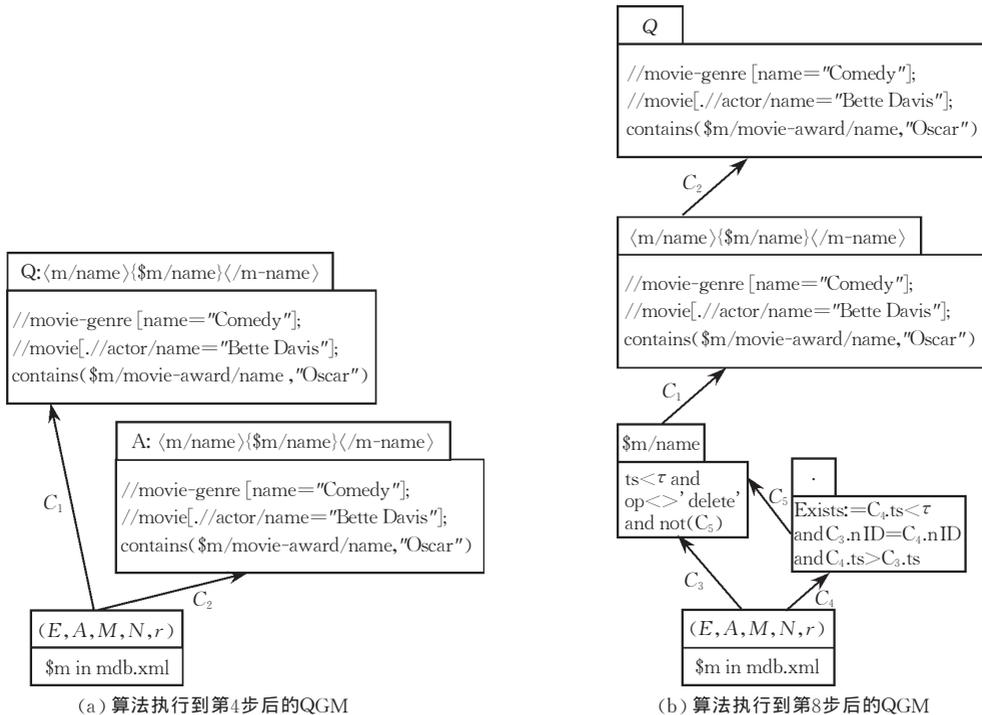


图 2 XQuery 查询的审计过程

$\Leftrightarrow \exists$  最小关键子树  $d_1$ , 同时  $\exists$  子树  $d_2$  和子树  $d_3$ , 使得  $P_Q(d_1 \times d_2) \wedge P_A(d_1 \times d_3)$ ;

$\Leftrightarrow \exists$  最小关键子树  $d_1$ , 同时  $\exists$  子树  $d_2$  和子树  $d_3$ , 使得  $\{d_1 \times d_2 \times d_3\} \in (\sigma_{P_A}(\sigma_{P_Q}(D_1 \times D_2 \times D_3)))$ .

$\Leftrightarrow (\sigma_{P_A}(\sigma_{P_Q}(D_1 \times D_2 \times D_3))) \neq \emptyset$ . 证毕.

### 3.3 审计算法

算法 1. 对 XQuery 查询的审计.

1. 为审计查询 AQ 创建空查询图模型 QGM(Query Graph Model);
2. 从查询日志中选择  $R_N \supseteq A_{ON}$  的候选查询 XQuery Q;
3. 添加 XQuery 查询 Q 到 AQ;
4. 添加审计 A 到 AQ;
5. 将 A 改为对候选查询 XQuery Q 的结果进行审计;
6. 用 XQuery 查询 Q 的标识来代替 A 的审计列表;
7. 用  $\tau$  时刻的文档树模型  $D^\tau$  来代替原来的文档树模型 D.
8. 输出审计结果.

例 2 的审计过程如图 2 所示, 执行到第 4 步后所产生的 QGM 如图 2(a) 所示, 执行到第 8 步时所得到的 QGM 如图 2(b) 所示.

## 4 XQuery 查询 Q 的推理审计

一般来说, 用户在得到查询结果的同时, 都能利用一般知识进行推理访问. 下面的例子表示了推理访问的情形.

例 3. 如图 3 所示的 XML 文档 (hospital.xml<sup>[1]</sup>), 它表示病人和医生的相关信息, “pname” 表示病人

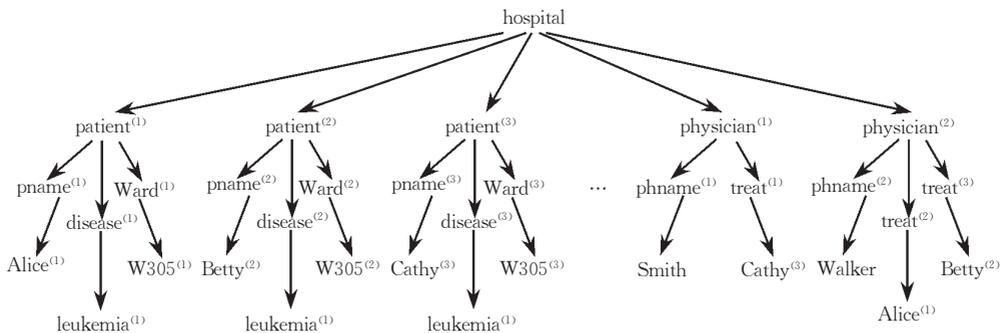


图 3 XML 文档 hospital.xml

### 4.1 一般的推理知识

提交查询的用户能使用各种 XML 约束进行推理, XML 约束一般来说可以表示成语义约束: “条件  $\rightarrow$  事实” 的形式, 它是 XML 文档上结点之间的关系, 即当条件在 XML 文档上成立时, 那么事实也同时成立, 通常在 XML 文档上有以下几类形式的约束<sup>[1,7]</sup>:

父子约束. 表示为  $//p \rightarrow //p/p'$  的形式, 即父结

点的名字, “ward” 表示病房号, “disease” 为病人所患疾病, “pname” 和 “treat” 则表示医生的名字及该医生治疗的病人, 现在假设查询和审计分别为

XQuery 查询 Q:

```
for $h in document("hospital.xml"/patient[ward = "W305"])
```

```
where contains($h/patient/pname, "Cathy")
```

```
return <h-disease>{$/disease}</h-disease>
```

Audit 审计 A:

```
audit <h-disease>{$/disease}</h-disease>
```

```
for $h in document("hospital.xml")
```

```
where contains($h/patient/pname, "Alice")
```

查询 Q 访问了住在 “W305” 病房的病人 “Cathy” 所患疾病, 由于病人 “Alice” 所患疾病对外界是保密的, 审计则想知道是哪些查询访问了病人 “Alice” 所患疾病, 从第三部分一般审计所表达的思想, 虽然有  $R_N \supseteq A_{ON}$ , 即查询 Q 为候选查询, 但  $(\sigma_{P_A}(\sigma_{P_Q}(\text{hospital.xml}))) = \emptyset$ , 所以查询 Q 不是可疑查询. 而在现实中, 如果用户知道 “Alice” 也住在 “W305” 房间, 住在同一房间的病人通常都患有相同的疾病, 那么用户在提交查询 Q 的同时, 实际同时也访问了 “Alice” 所患疾病的信息. 而前面的审计却无法对这样的查询进行审计, 常规的审计结果会表明: XQuery 查询 Q 没有访问 Alice 所患疾病. 显然, 这不是所希望的审计结果, 而希望审计具有基本的推理审计能力, 能将那些间接访问审计表达式所指定特定信息的查询也审计出来.

点  $p$  一定有子结点  $p'$ , 如例 3 文档中有约束  $C_1: //patient \rightarrow //patient/pname$ , 其含义为每一位病人都有自己名字.

祖孙约束. 表示为  $//p \rightarrow //p//p'$  的形式, 即祖先结点  $p$  一定有子孙结点  $p'$ , 例如例 3 文档中有约束  $C_2: //patient \rightarrow //patient//disease$ .

函数依赖. 表示为  $p/p_1 \rightarrow p/p_2$  的形式,  $p_1$  和  $p_2$

为文档上有限非空路径的子集,对任意匹配  $p/p_1$  的两棵子树  $t_1$  和  $t_2$ ,如果它们在  $p_1$  上的值相等,那么这两棵子树在匹配  $p/p_2$  子树时, $p_2$  的值也是相等的.例如例 3 文档上有约束  $C_3://patient/ward \rightarrow //patient/disease$ ,即如果两棵子树在  $//patient/ward$  上的值相等,那么这两棵子树在  $//patient/disease$  上的值也相等,住在同一病房里的病人患有相同的疾病.

有了这些约束  $C$  后,给定一个查询结果,用户就能对查询结果利用 XML 约束进行推理,得到比

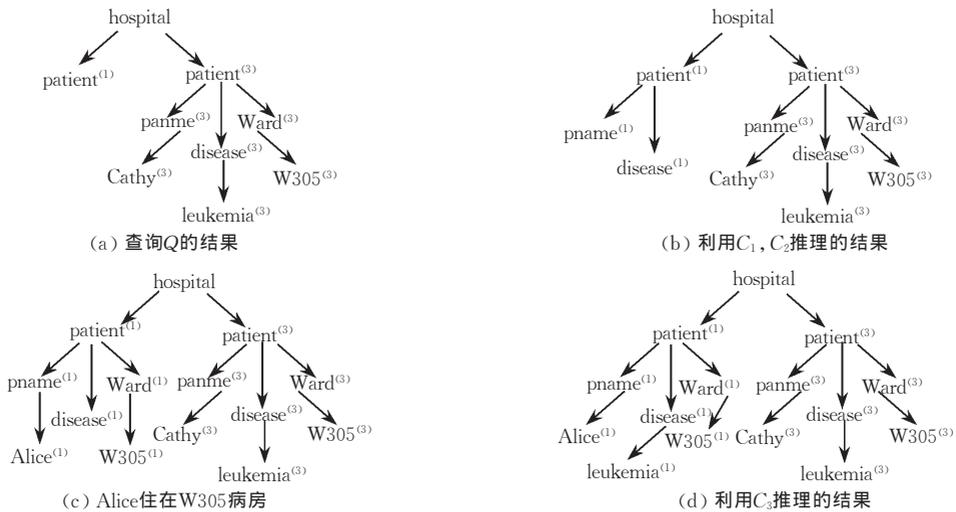


图 4 对 XQuery 查询进行推理的结果

**定义 7.** 等价映射文档<sup>[8]</sup> 给定两个 XML 文档  $D_1$  和  $D_2$ ,如果这里把  $D_1$  当作一个 XQuery 查询,称从查询  $D_1$  到文档  $D_2$  的映射为文档映射, $D_2$  映射包含  $D_1$ . 如果这两个文档相互映射包含,此时就说这两个文档是等价映射文档.

**定理 3<sup>[8]</sup>.** 给定一个 XML 文档  $D$  的一部分  $P$  和约束集合  $C = \{C_1, C_2, \dots, C_k\}$ ,存在唯一的文档  $M$ , $M$  由  $P$  按某种顺序使用约束推理得到,其它按任意顺序用约束对  $P$  推理得到的文档都映射包含于  $M$  之中.

把这样由  $P$  使用约束  $C$  进行推理而得到的文档  $M$  称为最大推理文档,记为  $M = C_{\max}(P)$ . 给定一个 XML 文档的一部分  $P$  和约束  $C$ ,使用 CHASE<sup>[7]</sup> 算法,就能得到最大推理文档  $M$ ,算法的基本思想是从  $C$  中任意选取约束对当前文档进行推理,如果推理得到的结果与原文档不是等价映射文档,则选取其它约束对结果文档继续进行推理,直到利用所有约束推理得到的新的结果文档与原来的结果文档为等价映射文档为止.

原来查询结果更多的信息.下面根据前面的查询给出一个例子,看推理怎样进行的.

**例 4.** 图 4(a) 为例 3 中查询所得到的结果,利用约束  $C_1, C_2$  就可以推理得到图 4(b) 所示的结果,而如果用户知道 Alice 住在 W305 病房(图 4(c)),就可以利用函数依赖约束  $C_3$  推理得到图 4(d) 所示的结果.在推理的过程中,用户运用不同顺序约束进行推理,所得到的推理结果也是不同的<sup>[8]</sup>,有些推理结果可能含有相同的信息量.

## 4.2 XQuery 查询推理审计方法

由于用户能够利用 XML 文档上的约束对查询结果进行推理访问,所以在审计时,必须扩大审计的范围,原来候选查询和可疑查询的定义不再适用于推理审计,下面对候选查询和可疑查询重新进行定义.

**定义 8.** 候选查询(candidate query). 对 Xquery 查询  $Q$  和审计  $A$ ,当且仅当  $C_{\max}(R_N) \supseteq A_{ON}$  时, $Q$  为候选查询.这里的  $C_{\max}(R_N)$  表示用户使用约束集  $C$  对查询  $Q$  中的结果集  $R_N$  进行推理,所得到的最大推理文档中所包含的结果集.

**定义 9.** 可疑查询(suspicious query). 对审计表达式  $A$  而言,候选查询  $Q$  为可疑查询,如果  $A$  和  $Q$  有相同的最小关键子树  $T$ ,或者在用户使用约束集  $C$  对查询结果  $Q$  经推理后所得到的最大推理文档  $M = C_{\max}(Q)$  与  $A$  有相同的最小关键子树  $T$ ,即  $(MinCri(T, Q)) \wedge (MinCri(T, A)) \neq \emptyset$ ,或者  $(MinCri(T, M)) \wedge (MinCri(T, A)) \neq \emptyset$ .

在增加推理审计功能后,推理审计步骤就应该在原来审计的基础之上(审计步骤的第 4 步之后)加

上推理审计步骤: 审计推理, 有些查询在经推理之后, 能访问查询结果之外的信息, 可能是潜在的可疑查询; 还有些查询, 自身经推理后并不一定能访问查询结果之外的信息, 但与其它查询相结合后, 却能访问所有这些查询结果之外的信息, 从而也是可疑查询, 即  $(MinCri(T, Q)) \wedge (MinCri(T, A)) \neq \emptyset$ , 或者  $(MinCri(T, M)) \wedge (MinCri(T, A)) \neq \emptyset$ . 这里由于推理的结果有可能是用户利用多个查询结果推理得到, 所以  $M = C_{\max}(Q_1, Q_2, \dots, Q_m)$ , 表示使用约束集对多个查询结果集的并进行推理所得到的最大推理文档.

**定理 4.**  $D_1, D_2, D_3$  为 XML 文档树模型,  $Q = \mathcal{R}_{ON}(\sigma_{P_Q}(D_1 \times D_2))$  为 XQuery 查询, 对审计表达式  $A = \mathcal{A}_{ON}(\sigma_{P_A}(D_1 \times D_3))$  而言, 当满足下列条件时, XQuery 查询  $Q$  为可疑查询.

- (1)  $C_{\max}(R_N) \supseteq A_{ON}$ ;
- (2)  $(\sigma_{P_A}(\sigma_{P_Q}(C_{\max}(Q)))) \neq \emptyset$ .

**证明.** 满足条件(1), 由前面关于候选查询的定义可知, XQuery 查询  $Q$  为候选查询.

按照可疑查询的定义,

$susp(Q, A) \Leftrightarrow \exists$  最小关键子树  $d \in C_{\max}(Q)$ ,  $d$  同时为  $C_{\max}(Q)$  和  $A$  的最小关键子树, 即有  $(MinCri(d, C_{\max}(Q))) \wedge (MinCri(d, A))$ .

$\Leftrightarrow \exists$  最小关键子树  $d$ , 使得  $P_Q(d) \wedge P_A(d)$ .

$\Leftrightarrow (\sigma_{P_A}(\sigma_{P_Q}(C_{\max}(Q)))) \neq \emptyset$ . 证毕.

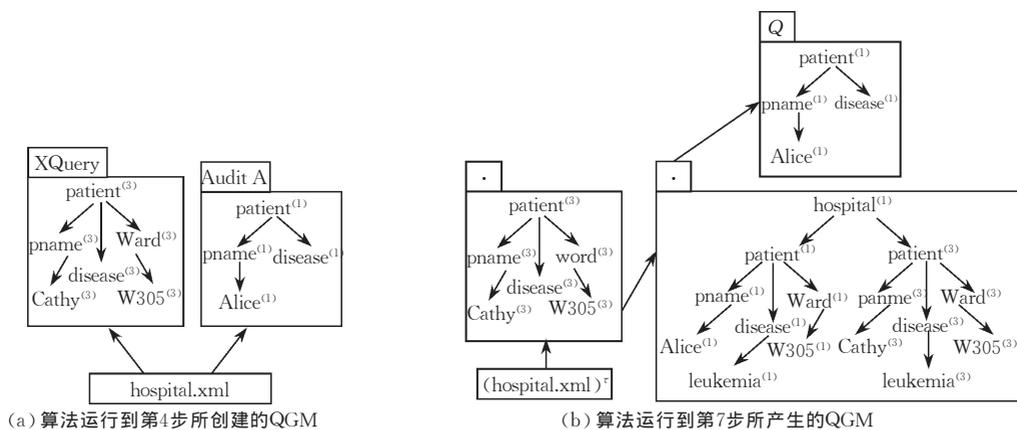


图 5 XQuery 查询的推理审计过程

## 5 审计及推理审计框架

实现对 XQuery 查询的审计, 并在此基础上, 利用 XML 上存在的各种约束, 使审计具有一定的推理能力, 最终所得到的审计模型和推理审计模型如图 6 所示. 在推理审计的过程中, 为了能够审计某

## 4.3 XQuery 查询推理审计算法

**算法 2.** 对 XQuery 查询的推理审计.

1. 为审计查询  $AQ$  创建空查询图模型 QGM (Query Graph Model);
2. 从查询日志中选择  $C_{\max}(R_N) \supseteq A_{ON}$  的候选查询 XQuery  $Q$ ;
3. 添加 XQuery 查询  $Q$  到  $AQ$ ;
4. 添加审计  $A$  到  $AQ$ ;
5. 将  $A$  改为对候选查询 XQuery  $Q$  的推理结果  $M = C_{\max}(Q)$  进行审计;
6. 用 XQuery 查询  $Q$  的标识来代替  $A$  的审计列表;
7. 用  $\tau$  时刻的文档树模型  $D^\tau$  来代替原来的文档树模型  $D$ .

当然, 用户在利用约束集  $C$  对查询结果进行推理时, 可能结合多个查询的查询结果来推理访问, 所以在审计的过程中, 也应该结合多个查询的结果来推理审计, 在推理审计系统实现的过程中, 按照用户的 ID 对查询进行划分, 将具有相同用户 ID 的查询结果放在一起进行审计.

## 4.4 XQuery 查询推理审计举例

**例 5.** 假设查询和审计仍为例 3 中所示的 XQuery 查询  $Q$  和审计  $A$ , 现在要对其中的查询利用 XML 约束  $C_1, C_2, C_3$  对查询  $Q$  进行推理审计, 推理审计所产生的 QGM 如图 5 所示. 这里对查询和审计用树型结构来表示, 并假定提交查询的用户在推理的过程中知道“Alice”住在“W305”病房.

些非法用户相互交换多个查询的查询结果进行推理访问, 需要将所有的查询结果集并在一起, 然后运用 XML 文档上所存在的各种约束, 对这个并集进行推理, 得到最后的最大推理文档, 然后对这个最大的推理文档进行审计. 但这是很困难的, 有以下两个方面的原因: (1) 如果最大推理文档使审计条件为真, 将返回所有查询的 ID, 从而使得某些并不属于审计范

围的查询也被当作是审计结果。(2)随着查询数的不断增加,审计的工作量会越来越大,最终会使审计无法实现.因此,在实际的实现过程中,不仅按审计表达式中的 during 子句对查询按时间段进行划分后审计,而且按提交 XQuery 查询的用户 ID 进行划分后再审计,主要审计各单个用户利用历史查询信息所进行的推理访问.

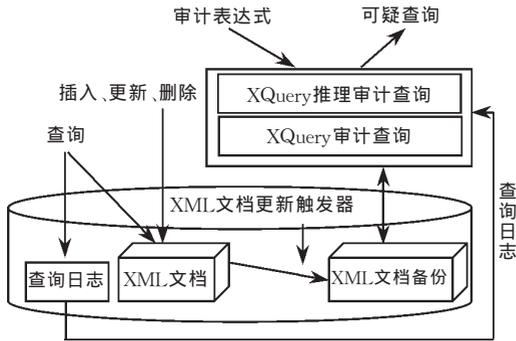
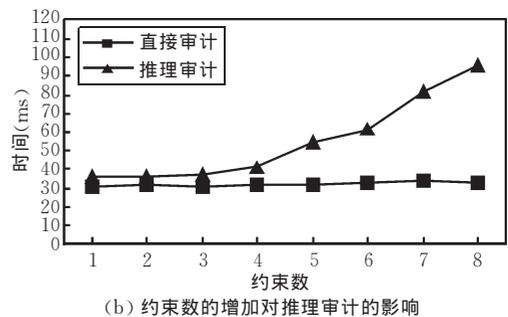
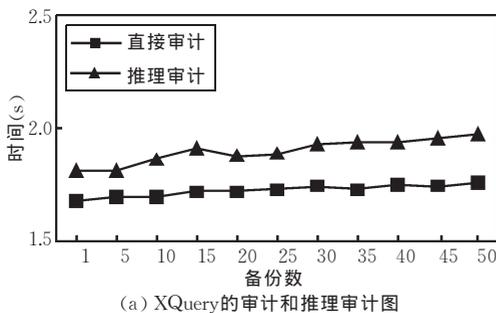


图 6 XQuery 查询的推理审计模型

## 6 实 验

实验结果旨在评价对 XQuery 查询进行审计的



(a) XQuery 的审计和推理审计图

(b) 约束数的增加对推理审计的影响

图 7 审计和推理审计实验结果

## 7 结 语

基于 Agrawal<sup>[5]</sup> 所提出的审计方法,提出了对 XQuery 查询进行审计的审计方法、算法和审计查询图模型(QGM),然后利用 XML 文档上存在的各种约束,使对 XQuery 的审计具有基本的推理能力,最终给出了 XQuery 查询的审计框架和推理审计框架. XQuery 审计和推理审计能够审计出任意直接或间接访问了特定数据的查询,且保持了原有审计模型快速、准确、细粒度等特点.对 XQuery 的审计属于安全数据库中的一部分,这种审计和推理审计的实现在提高数据安全性的同时,在很大程度上提高了数据的可用性.当然,在实际的推理审计实现过

效率,在加上推理审计功能后对原审计功能所产生的影响,特别是当推理导致敏感信息泄露的各种约束数增加时对推理审计所带来的影响.实验是在处理器为 P4 1.8GHz,内存为 512MB,操作系统为 Windows XP,缓冲池大小为 256MB 的环境下完成的.数据集为文献[1]中所提供的数据集:dblp.xml 和 course\_washington.xml,完成两组实验.第一组针对 dblp.xml 上存在的函数依赖,比较审计和推理审计的效率.第二组针对 course\_washington.xml 上存在的五条父子约束、两条祖孙约束和函数依赖,考察随着约束数对推理审计带来的影响.实验结果如图 7 所示.图 7(a)表明,在备份的数据上建立组合索引(主键和时间)后,备份数的增加对审计和推理审计基本没有产生影响,在利用函数依赖进行推理审计时,效率有所下降,但总的下降幅度不大,在 10%左右,完全可以接受.图 7(b)表示当约束数增加时推理审计的效率,在约束数增加到 8 条时,效率下降到了近 300%,考虑到审计不是实时审计的,且在实际的数据集中,约束数较少,这样的审计结果处在可以接受的范围.

程中,按照提交查询的用户 ID 进行了划分,只能审计单个用户利用历史查询进行推理访问的查询,而没有实现对多个用户相互交换查询结果后的推理访问进行审计,这将是本课题今后的研究工作重点.

## 参 考 文 献

- 1 Yang X., Li C.. Secure XML publishing without information leakage in the presence of data inference. In: Proceedings of the 30th VLDB conference, Toronto, Canada, 2004, 96~107
- 2 Abadi M., Warinschi B.. Security analysis of cryptographically controlled access to XML documents. In: Proceedings of the ACM PODS, Baltimore, Maryland, 2005, 108~117
- 3 Miklau G., Suciu D.. Controlling access to published data using cryptography. In: Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003, 898~909

- 4 Rizvi S. , Mendelzon A. , Sudarshan S. , Roy P. . Extending query rewriting techniques for fine-grained access control. In: Proceedings of the ACM SIGMOD, Paris, France, 2004, 551~561
- 5 Agrawal R. , Bayardo R. , Faloutsos C. , Kiernan J. , Rantzaou R. , Srikant R. . Auditing compliance with a Hippocratic database. In: Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004, 516~527
- 6 Agrawal R. , Kiernan J. , Srikant R. , Xu Y. . Hippocratic databases. In: Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002, 143~154
- 7 Jagadish H. V. , Lakshmanan L. V. S. , Scannapieco M. , Srivastava D. , Wiwatwattana N. . Colorful XML: One hierarchy isn't enough. In: Proceedings of the ACM SIGMOD, Paris, France, 2004, 251~262
- 8 Yang X. , Li C. . Secure XML publishing without information leakage in the presence of data inference (Full Version). UC Irvine: Technical Report, 2004
- 9 Fan Wen-Fei, Chan Chee-Yong, Garofalakis M. . Secure XML querying with security views. In: Proceedings of the ACM SIGMOD, Paris, France, 2004, 587~598



**YAN He-Ping**, born in 1970, Ph.D..

His research interests include database security, XML, and data mining.

**LIU Bing**, born in 1978, Ph. D. . His research interests include data mining, data warehouse, database, and depend-

able system and networks.

**WANG Wei**, born in 1970, professor, Ph. D. supervisor. His research interests include database, secure database, data mining, and XML.

**SHI Bo-Le**, born in 1936, professor, Ph. D. supervisor. His research interests include database and knowledge database, data mining, digital library, and security database.

## Background

The work is supported by the National Natural Science Foundation of China under grant No. 60303008: "Research on Technology of Storing and Querying Cluster Data", No. 69933010: "Research on Key Technology of Digital Library"; and also supported by the National High Technology Research and Development Program (863 Program) of China under grant No. 2002AA4Z3430: "Web Service Based New Database Technology", supported by the industrialization development of special funds from Shanghai Information Committee. The research group has done a lot of research works in the past two years in this field including the architecture of

multi-level secure database system, authentication and encrypted data storage, and several papers have been published in the international conferences.

The contribution of this paper is as following: since the XML data is different from the general data in storage, query, and constraints among the data, the paper mainly focuses on how to audit and inference audit the XQueries. The carefully designed experiment shows the effectiveness and efficiency of the proposed XQuery audit and inference audit framework.