基于切片的OLAP动态推理控制研究

周彩霞 陈启买
(华南师范大学计算机学院 广州 510631)

摘 要 针对OLAP系统存在的数据仓库敏感信息泄漏的问题,及已有的推理研究都是以数据立方体为粒度,细粒度的切片推理仍然存在的问题,提出了以切片为推理单元的推理控制方法。该方法将推理粒度细化到切片,使每次查询生成对应的切片格,根据格的依赖关系判定是否存在推理通道,实现动态地防止单切片的推理,提高了敏感信息的保护力度。

关键词 数据仓库,在线分析处理系统,数据立方体,切片,推理控制
中图法分类号 TP311.1 **文献标识码** A

Slice-based Method for Dynamic Inference Control in OLAP

ZHOU Cai-xia CHEN Qi-mai
(School of Computer, South China Normal University, Guangzhou 510631, China)

Abstract Aiming at the problem of sensitive information leakage in data warehouse and OLAP system, and the existing inference control methods are based on date cube ignoring the fine-grained slice reasoning. This paper presents an inference control method based on slice. This method refines inference granularity to slice and deals with each query corresponding to slice grid, which can prevent a single slice of reasoning and increases efforts to the protection of sensitive information according to the judgment of the slice grid dependencies' inference channel.

Keywords Data warehouse, OLAP system, Data cube, Slice, Inference control

1 引言

决策支持系统如在线分析处理(On-Line Analytical Processing,OLAP)系统在工业应用中日益重要,这类系统需及时响应海量数据的查询和统计。尽管OLAP系统拥有性能强大、不断完善的访问控制技术,但还是存在各种安全推理威胁所产生的有关数据仓库敏感信息泄露的安全问题[1-3,6]。

国外Lingyu Wang等人[4]首先比较详细地分析了OLAP系统中常用的查询组合有可能对数据隐私带来严重的威胁,提出通过限制用户的查询来预防推理通道的产生,但也提出了合法用户经OLAP的组合查询,可以推理得到相关敏感信息的问题[5];接着展开了将推理集细化的工作,由方体作为敏感对象细化到的所有单元均为敏感对象[6-8]。但是其没有考虑用户历史查询组合产生的推理威胁,也没有解决细粒度的切片推理控制问题。国内,文献[9]提出了先预防 m 维推理,然后清除一维推理的"预防-清除"静态推理控制方法,同样避开了历史查询组合所产生的推理威胁,但静态的方法实用性不强。文献[10]针对文献[6]的静态推理的不足实现了动态推理控制方法,但仍然没有解决细粒度推理控制问题。

针对上述已有推理方法的不足,且国内外对推理控制的研究都没有新的进展,本文针对已有研究的推理粒度不够细的问题,提出了一种细粒度的基于切片的OLAP动态推理方法,该方法在切片级别上来定义敏感信息,并能在查询时及时判定是否存在切片推理,可以有效地提高数据仓库的安全性和实时性。本文的优势体现在:(1)把推理粒度细化到切片,可以有效提高切片级别的组合推理,提高了OLAP决策系统的安全性;(2)动态推理可以保证OLAP系统的实时性。

2 基于切片的推理控制

数据立方体是OLAP系统的基本数据模型,最早是为了支持一般的OLAP任务而被Gray等人[11]作为一种SQL操作提出的,此后该模型被广泛应用于OLAP系统和数据仓库中。数据立方体可由数据仓库的事实表导出,其由基本单元(cell)组成,包含各种维组合的度量数据。本文参照二维数据立方体的格结构[11],将以切片格的形式来研究切片级别的推理。图1是一个简单的二维(时间维timeDim和员工维employeeDim)数据立方体,时间维有2个维属性:quarter、year;员工维有3个属性:employee、department、branch。

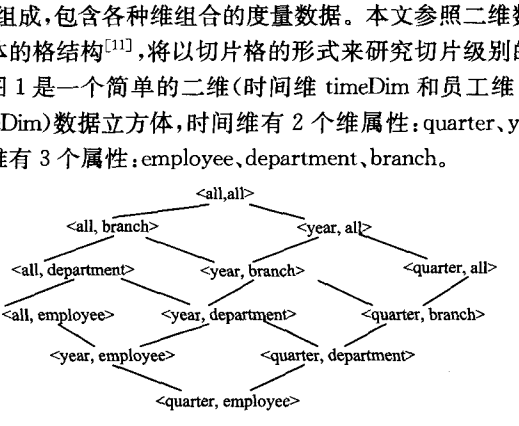


图1 二维数据立方体的格结构

本文受广东省科技计划基金项目(2009B010800036),广东省教育科研基金项目(BKYBJG20060235)资助。
周彩霞(1986—),女,硕士生,主要研究方向为数据库与数据挖掘,E-mail:zcx_202@163.com;陈启买(1965—),男,教授,主要研究方向为高性能数据库、数据仓库与数据挖掘。

从图 1 的二维数据的核心方体〈quarter, employee〉出发，公司维属性中的员工属性被分为 $employee_1$ 与 $employee_2$ 切片，他们分别对应的后代切片属性是 $employee_1 \rightarrow department_1 \rightarrow branch_1 \rightarrow all_1$ 和 $employee_2 \rightarrow department_2 \rightarrow branch_2 \rightarrow all_2$。图 2 就是根据第 1 种依赖关系生成的格结构。

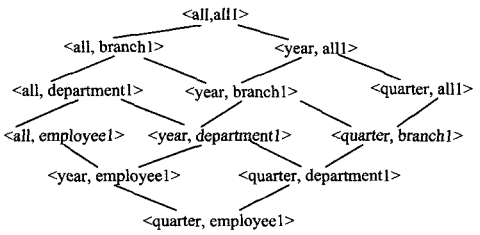


图 2　切片格结构

2.1　相关概念的形式化定义

定义 1(格)　数据立方体 Γ 中每个维度属性之间都存在偏序的依赖关系，本文把这种依赖关系看作依赖格，简称格，用符号“≤”表示依赖关系，〈Γ, ≤〉表示格，如员工维中属性级别依赖关系有 $employee \leq department \leq branch \leq all$。

定义 2(取值函数 *GetValue*())　$GetValue(AttrID_j^i) = \{s : s \in AttrID_j^i, value\}$ 表示在属性级别 $AttrID_j^i$ 的所有取值，其中，$AttrID_j^i$ 表示数据立方体 Γ 中 Dim_i 维层次中的第 j 属性级别。

例 1　令 $Dim_1 = employeeDim$，则 $AttrID_2^1 = department$；当 $Value = \{department_1, department_2\}$ 时，$GetValue(AttrID_2^1)|_{value=Value}$ 表示取员工维对部门属性为 $department_1, department_2$ 的值。

定义 3(属性规约函数 *AttrIDReduce*())　$AttrIDReduce(AttrID_j^i) = \{AttrID_{j|GetValue(AttrID_j^i)|}^i\}$ 表示在任意立方体 Γ 中，对 $AttrID_j^i$ 存在依赖关系的属性进行规约。其中，$AttrID_j^i$ 表示数据立方体 Γ 中 Dim_i 维层次中的第 j 属性级别。

例 2　当 $i=1$ 时，$Dim_1 = employeeDim$，$AttrID_2^1.value = \{department_1, department_2\}$，则 $GetValue(AttrID_2^1) = \{department_1, department_2\}$，$AttrIDReduce(AttrID_2^1) = \{employees_1 \leq department_1 \leq branch_1 \leq all_1, employee_2 \leq department_2 \leq branch_2 \leq all_2\}$。

定义 4(切片)　在任意数据立方体 Γ 中，对第 i 维 Dim_i 进行切片，则切片表示为：$Slice(\Gamma)|_j^i = \{\langle Dim_1, Dim_2, \cdots, Dim_{i-1}, GetValue(AttrID_j^i), Dim_{i+1} \cdots Dim_k\rangle\}$，其中 $1 \leqslant i \leqslant k$，$1 \leqslant j \leqslant |Dim_i, Hier.Lever|$。

例 3　在员工数据立方体 Γ 中，令 $Dim_1 = employeeDim$，$AttrID_2^1 = department$，当 $Value = \{department_1\}$ 时，切片 $Slice(\Gamma)|_2^1 = \{\prod_m^n \langle timeDim.Hier.Lever_t, GetValue(department)|_{value=Value}\rangle\}$，其中 $m = \{t \mid t=1\}$，$n = \{t = |timeDim.Hier.Lever|\}$，表示所有 *department* 属性级别上的所有切片：〈*quarter*, $department_1$〉，〈*year*, $department_1$〉，〈*all*, $department_1$〉。

定义 5(切片方体)　在任意数据立方体 Γ 中，存在切片 $Slice(\Gamma)|_j^i$，当 m 和 $Value$ 取值唯一时，切片 $Slice(\Gamma)|_j^i$ 表示唯一的切片方体 $SliceCuboid(m, Value)$。

例 4　如在切片定义中，当 $(m, Value) = (1, department_1)$ 时，$SliceCuboid(1, department_1) = \langle quarter, department_1\rangle$。

定义 6(切片格)　在任意数据立方体 Γ 中，对第 i 维 Dim_i 进行切片，切片可表示为 $Slice(\Gamma)|_j^i$，其中 j 表示维 Dim_i 中第 j 级别属性，根据维级别的偏序关系和依赖关系存在切片 $Slice(\Gamma)|_{jh}^i$ 和切片 $Slice(\Gamma)|_{1<j}^i$，令 $Slice(\Gamma)|^i = Slice(\Gamma)|_{1j}^i \cup Slice(\Gamma)|_j^i \cup Slice(\Gamma)|_{jh}^i$，则定义切片格 $\Gamma^i = \langle Slice(\Gamma)|_j^i, \leq\rangle$。

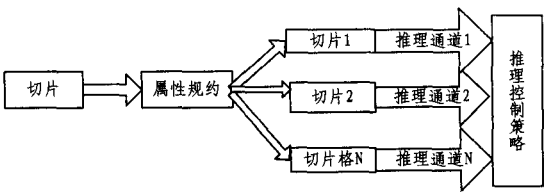
定义 7(授权函数 *Authorization*())　设客体 $Object = \{SliceCuboid\}$，主体 $Subject = \{Role, User\}$，则授权函数 $Authorization() = (Object, Subject, allowed/denied)$，表示 *Subject* 具有在 *Object* 上的安全信息 *allowed* 或者 *denied*。

定义 8(敏感对象)　敏感对象 $SensitiveObject = (Object, denied, Subject)$，指对于某个主体，它是不可以访问的，否则会导致信息泄露。对应的非敏感对象 $\overline{SensitiveObject}$ 表示为 $\overline{SensitiveObject} = (Object, allowed, Subject)$，则表示对于某个主体，它是可以安全访问的。

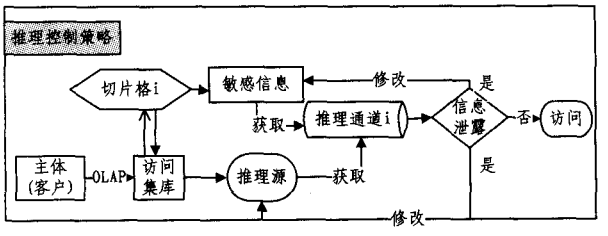
定义 9(库)　在任意数据立方体 Γ 中，切片格 $\Gamma^i = \langle Slice(\Gamma)|_j^i, \leq\rangle$，库的形式为 $Based = (\Gamma^i, SC_{request}^i, Source^i, Target^i, DynamicTarget^i)$ 表示同一主体在访问不同切片格 Γ^i 时的访问轨迹 $SC_{request}^i$ 集合、动态变化的可访问源切片方体集合 $Source^i$、不可访问的目标切片方体集 $Target^i$ 及动态目标方体集合 $DynamicTarget^i$。

2.2　切片推理控制算法模型

本文的切片推理是在切片格的结构上展开的。该方法的主要思想是将方体中的每一个可能的切片看作一个新的方体，这种新的方体间又建立格的依赖关系，最后采用基于切片格的推理控制原理实现切片的动态推理控制。如图 3(a)及图 3(b)所示，对所有切片进行属性规约后形成不同的 n 个切片格，每个切片格相互独立，并存在各自的推理通道，最后采取类似的推理策略解决其推理问题。



(a) 推理通道图



(b) 推理控制策略图

图 3

3　OLAP 切片推理算法

基于上述切片推理控制算法模型，可以得出单切片格推理控制算法的思想是：先从当前主体获得其敏感信息，并根据

这些敏感信息生成相应的切片格。根据当前主体的查询语句，获取其切片值，然后对这些切片值进行属性规约，形成新的切片格；判定当前查询所在的切片格是否在主体所属的敏感切片格中：若是，则判定当前查询所属的切片格是否已经访问过，如果访问过则从库中获取其当前的目标切片方体集与源切片方体集，如果没有访问过则设其库中对应的元素为空；否则，判定当前查询切片与主体所属的敏感切片格中的切片方体的关系，如果当前查询切片包含主体所属的敏感切片格中的切片方体中的切片方体部分敏感切片，则阻塞当前查询。当且仅当查询切片不包含部分或者全部敏感切片时，才不阻塞其访问，并更新库中对应的元素。

3.1 算法实现

单切片格推理控制算法的描述如下：

输入：$SC_{request}$；

输出：prevent() or submit()

初始化：

1. 是一个数据立方体，Base. Add($SC_{request}$) //将所查询的切片加入到库中。

2. prevent() {

if ∃inference { prevent access; DynamicTarget^i ∪ = { sc: sc ≤ $SC_{request}$ }; $SC^i_{request}$ \= $SC_{request}$; Target^i ∪ = DynamicTarget^i; Base. Update((Γ^i, $SC^i_{request}$, Source^i, Target^i, DynamicTarget^i); }}

3. submit() {

if noninference then{submit to OLAP system; $SC^i_{request}$ ∪ = $SC_{request}$; Base. Update((Γ^i, $SC^i_{request}$, Source^i, Target^i, DynamicTarget^i);

4. get SensitiveObject by Subject;

5. format all slice-lattice $\Gamma^{Subject}$ by SentiveObject; get $Target^{Subject}$ = Targe(SensitiveObject; get $Source^{Subject} = \Gamma^{Subject} \setminus Target^{Subject}$; Base. Add ($\Gamma^{Subject}$, $Source^{Subject}$, $Target^{Subject}$) by each slice-lattice in $\Gamma^{Subject}$;

算法主体：

Begin;

Step1 If $SC_{request} \in$ Base $Target^{Subject}$ then return prevent();

Step2 GetValue(AttrID^i_j) by $SC_{request}$; //获取当前访问的切片值

Step3 if $\forall SC_{request} \in \Gamma^{Subject}$ then goto Step4; else { if $\exists SC_{request} \in \Gamma^{Subject}$ then { if $\exists SC_{request} \cap$ Base. $Target^{Subject} = \Phi$; then{ ADR = AttrIDReduce(AttrID^i_j); Format all slice-lattice $\Gamma^{|ADR|}$ by ADR;

If $\Gamma^{|ADR|}$ in Base then goto Step4;

else if $SC_{request} \in \Gamma^{|ADR|}$ then { if i value is new //如果是初次访问的切片格

then{get SensitiveObject^i and $\overline{\text{SensitiveObject}^i}$ by Subject and Γ^i;

get Target^i = Target^i(SecsitiveObject^i); get Source^i = Γ^i \ Target^i;

Base. Add(Target^i, Source^i)//更新库，添加目标集和源集

return submit();}

else{set Target^i = Base. get(Target^i); set Source^i = Base. get (Source^i);}} }

else return prevent();}}

Step4 if(Apex Cuboid ∈ $Target^{base}$ or $Source^{base} = \Phi$ or $SC^{base}_{request} \cap Target^{base} \neq \Phi$ then return prevent();

Step5 if GLB($SC^{base}_{request}$) = Φ then goto Step8;

Step6 if $Target^{base} \cap$ Above(GLB($SC^{base}_{request}$)) ≠ Φ then return prevent();

Step7 if DR = DynamicRoot($SC^{base}_{request}$, SentiveObject^{base}) = Φ then goto Step8; else return submit();

Step8 if DB = DynamicBasic($SC^{base}_{request}$) = Φ then{ if |$SC^{base}_{request} \cap$ DB| > 1| then return prevent(); else return submit();} else return submit();

End

算法复杂度分析：设一个数据立方体有 m 维，每个维 Dim_i ($2 \leqslant i \leqslant m$) 中的级别数为 $length_{ij}$ ($j \geqslant 1$)，则其方体总数 $N = \prod_{2 \leqslant i \leqslant m, j \geqslant 1} (length_{ij} + 1)$，空间复杂性 T 取决于方体的总数，$T = O(N)$，时间复杂性取决于递归求解方体的祖先与后代关系时消耗的时间，$T = O(N!)$

3.2 实例说明

问题描述：设员工数据立方体 $\Gamma = \prod_{i=1}^{k} Dim_1$，其中 $Dim_1 = timeDim$ 和 $Dim_1 = employeeDim$。

初始化：

(1)客体 $Object_1 = \{SC_1 = \langle quarter, department_1 \rangle, SC_2 = \langle quarter, department_2 \rangle, SC_3 = \langle quarter, department_3 \rangle; Object_2 = \{\langle quarter, department_4 \rangle\}$；

(2)主体 $Subject = \{Role_1\}$；

(3)切片方体安全性：$SSC_1 = \{SC_1, Readdenied\}; SSC_1 = \{SC_2, Readdenied\}; SSC_1 = \{SC_3, Readdenied\}$；

(4)安全信息：$SecurityInfo = \{SSC_1, SSC_2, SSC_3\}$；

(5)授权 $Authorization(Object_1, Subject, SecurityInfo)$。

初始化部分：$SensitiveObject = \{sc: sc \in SC_1 \cup SC_2 \cup SC_3\}; \Gamma^{Subject} = \{\Gamma: \Gamma^1 \cup \Gamma^2 \cup \Gamma^3\}; Target^{subject} = \{t: t \in Target^1 \cup Target^2 \cup Target^3\}; Target^i = \{sc: sc \in \{sl: sl \leq SensitiveObjbect_i\}\}$，其中 $i = 1, 2, 3$；$Source^{Subject} = \{s: s \in Source^1 \cup Source^2 \cup Source^3\}, Source^i = \{sc: sc \in \Gamma^i \setminus Target^i\}$，其中 $i = 1, 2, 3$；$Base.Add\{\Gamma^{Subject}, Target^{subject}, Source^{Subject}\}$。

下面分别查询属于某切片格或者部分属于某些切片格和不属于切片格的实例进行说明：

(1)当查询的客体属于 $\Gamma^{Subject}$ 中的切片格时，有两种情况：

1)客体属于目标切片方体集中的元素时，直接阻塞当前查询。设当前查询 $SC^1_{request} = SC_1$，执行 Step1：$SC_{request} \in Base.Target^{Subject}$，直接阻塞当前查询，算法结束。

2)客体属于源切片方体集中的元素时，则可能产生推理通道：

a)直接后代。设第一次查询的客体 $SC^1_{request} = \langle quarter, branch_1 \rangle$，根据其切片格的依赖关系有 $SC^1_{request} \in \Gamma^1$（执行 Step3 转到 Step4），并且具有 $SC_1 \leq SC^1_{request}$）关系，易知，不能推理得到目标切片方体，此时更新库 $Base$ 中的 $Base.SC^1_{request} \cup = SC^1_{request}$；

b)兄弟关系。设第二次查询的客体 $SC^2_{request} = \langle year, department_1 \rangle$，根据其切片格的依赖关系有 $SC^2_{request} \in \Gamma^1$（执行 Step3 转到 Step4），并且具有 $Brother(SC^1_{request}, SC^2_{request} = true)$，执行 Step6 可得它们最大下界是敏感切片方体 SC_1，则

阻塞当前查询，并更新 $Base$ 中的 $Base.Target^1 \cup = \{SC^2_{request}, SC^{Base}_{request} \setminus SC^2_{request}\}$。

c)异切片格。设第三次查询的客体 $SC^3_{request} = \langle quarter, branch_2 \rangle$，根据其切片格的依赖关系有 $SC^3_{request} \in \Gamma^2$（执行 Step3 转到 Step4），并且具有 $SC_2 \preceq SC^2_{request}$ 关系，易知不能推理得到目标切片方体（执行 Step7），则此时更新库 $Base$ 中的 $Base.SC^{Base}_{request} \cup = SC^3_{request}$。

(2)当查询的客体不属于 $\Gamma^{subject}$ 中的切片方体时，则查询客体生成相应的所有切片格结构 $\Gamma^{|ADR|}$，判断其属性规约后有以下几种情况：

1)当前访问客体与库中的目标切片方体集有共同部分属性，为确保敏感信息的安全性，则阻塞当前查询。设当前查询是对（$department_1, department_4$）的切片，即 $SC^1_{request} = \langle quarter, (department_1, department_4) \rangle$，则执行 Step3，满足 $\exists SC^1_{request} \in \Gamma^{subject}$，并且 $\exists SC^1_{request} \cap Base.Target^{subject} \neq \phi$ 泄露目标切片方体，则阻塞当前查询。

2)当前访问客体与库中的目标切片方体集没有共同部分属性：

i. 设第一次查询 $SC^1_{request} = \langle quarter, (branch_1, branch_2, branch_4) \rangle$，显然 $SC^1_{request} \cap Base.Target^{subject} = \phi$（执行 Step3），此时执行属性规约操作 ADR，形成新属性 $Branch^{|ADR|}{}_1 = \{branch_1, branch_2, branch_4\}$，$Branch^{|ADR|}{}_2 = \{branch_3\}$，根据 employeeDim 维层次级别的依赖关系生成相应的 $employee^{|ADR|} \preceq department^{|ADR|} \preceq branch^{|ADR|} \preceq all^{|ADR|}$，对应的切片结构为 $\Gamma^{|ADR|}$，因为在切片格 $\Gamma^{|ADR|}$ 中未定义安全信息，所以任何查询都不会产生敏感信息泄露，则允许当前查询，并更新 $Base.SC^{|ADR|}_{request} \cup = SC^1_{request}$；

ii. 设第二次查询 $SC^2_{request} = \langle quarter, (department_1, department_2, department_4) \rangle$，显然 $\exists SC^2_{request} \cap Base.Target^{subject} \neq \phi$（执行 Step3），泄露目标切片方体，则阻塞当前查询，并更新库 $Base$ 中的 $Base.SC^{|ADR|}_{request} \cup = SC^2_{request}, SC^{Base}_{request} \setminus SC^2_{request}$。此时切片格 $\Gamma^{|ADR|}$ 已经存在敏感方体了，并会随着查询的不断增加而增多；

iii. 设第三次查询 $SC^3_{request} = \langle year, (department_1, department_2, department_4) \rangle$，显然 $SC^3_{request} \cap Base.Targe^{subject} \neq \phi$（Step3），并且 $\Gamma^{|ADR|} \in Base$，转到 Step4，Step5 执行，由于条件不成立则直接执行 Step6，因为 $GLB(SC^{Base}_{request}) = \langle year, (department_1, department_2, department_4) \rangle$，与 $Targe^{Base}$ 的交集不为空，则泄露敏感信息，阻塞当前查询，并更新 $Base.SC^{ADR}_{request} \cup = SC^3_{request}, SC^{Base}_{request} \setminus SC^3_{request}$。

综上列举的几种典型查询可知，在已定义的切片格 $\Gamma^{subject}$ 中存在推理通道的同时，在动态查询过程中形成的切片格 $\Gamma^{|ADR|}$ 也存在推理通道，上述算法能很好地发现其推理通道并阻塞该查询，达到信息安全的目的。

3.3 算法评价

(1)安全性。本算法的核心思想是实时地对查询进行推理控制，在切片格中，当主体的请求达到饱和状态时，目标切片方体集合源切片方体集是唯一确定的集，其中的元素不再变化，所以满足安全性的需求。(2)提高了访问有效性。首先本方法从方体级别进行推理控制出发，到更细粒度切片级进行推理控制，推理粒度上进行了细化；其次切片只是方体的部分属性值，从细节数据量上进一步提高了访问有效性。(3)可靠性。针对不同情况的查询，算法都能做出正确的阻塞或不阻塞动作，说明本方法具有可靠性。

结束语 本文着重讨论 OLAP 系统中存在的切片为通道的推理控制问题，提出以切片为推理单元的推理控制方法。该方法以提高敏感信息的保护力度为目的，防止产生以切片为单元的细粒度推理。本文的研究工作基于切片查询的动态推理控制，在提高系统安全性的同时牺牲了一定的在线响应时间，在线响应性能方面很难和静态推理方法比较，所以在今后我们将结合静动态推理控制各自的优势做进一步的研究；另外，本文提出的方法没有考虑多切片的串谋推理威胁，所以如何防止多切片串谋推理威胁也是今后研究的一个重点。

参考文献

[1] Wang L, Li Y, Wijesekera D, et al. Cardinality-Based Inference Control in Data Cubes[J]. Journal of Computer Security, 2004, 12(5): 655-692

[2] Zhang Da-yong, Zeng Yong, Wang Ling-yu, et al. Modeling and evaluating information leakage caused by inferences in supply c hains[J]. 2011, 62(3): 351-363

[3] Wang Ling-yu, Lu H, Deng R H. Practical Inference Control for Data Cubes[C]// Proceeding of IEEE Transactions on Dependable and Secure Computing. Canana: IEEE Computer Society, 2006: 115-120

[4] Wijesekera L D, Jajodia S. Inferences in Data Cubes[J]. Berlin: Heidelberg, Springer-Verlag, 2002, 29(1): 37-51

[5] Wang Ling-yu, Wijesekera D, Jajodia S. Cardinality-Based Inference Control in Sum-OnlyData Cubes[J]. Berlin: Heidelberg, Springer-Verlag, 2002, 2502: 55-71

[6] Wang Ling-yu, Wijesekera D, Jajodia S. Parity-based Inference Control for Multi-dimensional Range Sum Queries[J]. Journal of Computer Security, 2007, 15(4): 417-445

[7] Wang Ling-yu, Li Ying-jiu. Precisely Answering Multidimensional Range Queries Without Privacy Breaches[C]// Proc. of ESORICS'03. Gjovik, Norway: [s. n.], 2003

[8] Wang Ling-yu, Wijesekera D, Jajodia S. Preserving Privacy in On-line Analytical Processing (OLAP)[M]. [S. l.]: Springer-Verlag, 2007

[9] 周海晴. 陈启买，刘海. 基于数据立方体的数据仓库安全控制[J]. 计算机工程，2010，36(10)

[10] 黄晓森，彭利宁，陈启买. 基于数据立方体的动态推理控制方法研究[J]. 计算机工程，2011，37(17)

[11] Gray J, Bosworth A, Bosworth A, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals[J]. Data Mining and Knowledge Discovery, 1997, 1(1): 29-53

[12] Donnellan T. Lattice Theory[M]. Pergamon Press, Oxford, 1968