软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(8):1872−1883 [doi: 10.3724/SP.J.1001.2011.03868]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

空间矢量数据细粒度强制查询访问控制模型及其高效实现*

张 妍[1,2], 陈 驰[1,2+], 冯登国[1,2]

[1](中国科学院 软件研究所 信息安全国家重点实验室,北京 100190)

[2](信息安全共性技术国家工程研究中心,北京 100190)

Fine-Grained Mandatory Query Access Control Model and Its Efficient Realization for Spatial Vector Data

ZHANG Yan[1,2], CHEN Chi[1,2+], FENG Deng-Guo[1,2]

[1](State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](National Engineering Research Center of Information Security, Beijing 100190, China)

+ Corresponding author: E-mail: chenchi821@gmail.com

Zhang Y, Chen C, Feng DG. Fine-Grained mandatory query access control model and its efficient realization for spatial vector data. *Journal of Software*, 2011,22(8):1872−1883. http://www.jos.org.cn/1000-9825/3868.htm

Abstract: To protect the spatial vector data, which is often in an irregular shape and distributed throughout multiple sensitive areas, the traditional mandatory access control model is extended and explained in this paper. This paper also proposes a fine-grained spatial mandatory query access control model—SV_MAC (spatial vector data mandatory access control model). Also, an AR+ spatial index tree technique is advanced, which combines the search of both spatial data and access control policies together to efficiently enforce the SV_MAC model in the course of spatial vector data searching. Experiment results shows that AR+ tree can not only provide fine-grained security protection for sensitive spatial vector data, but can also guarantee good user experience for GIS (geography information system) applications.

Key words: spatial vector data security; access control model; mandatory access control; R+ tree index; authorization model realization

摘 要: 针对敏感空间地理矢量数据形状不规则、跨多级敏感区域分布的特点,对传统的强制访问控制模型进行空间扩展,提出了一种细粒度的空间矢量数据强制查询访问控制模型 SV_MAC(spatial vector data mandatory access control model).并进一步将空间数据查询与安全策略检索相结合,提出了一种 AR+树(access R+树)索引结构,以在空间矢量数据查询过程中高效地实现 SV_MAC 授权判定.实验结果表明,AR+树在为空间矢量数据的检索提供不可绕过的细粒度安全防护的同时,保障了前台响应速率和用户体验.

关键词: 空间矢量数据安全;访问控制模型;强制访问控制;R+树索引;授权模型实现

中图法分类号: TP309 **文献标识码**: A

日益发展的 GIS(geography information system)地理信息系统技术在资源调查、灾害预测、国土管理等领

* 基金项目: 国家自然科学基金(61003228); 中国科学院知识创新工程领域前沿项目(ISCAS 2009-DR13)
收稿时间: 2009-11-11; 定稿时间: 2010-04-14

域的广泛应用,已将地理数据服务与人们的生活紧密相连.目前,很多地理空间数据可在互联网上快速甚至免费获得,不仅使人们能够清晰地观测国家任何区域的详细位置信息,而且能够监控个人的室外活动,给国家安全和个人隐私带来严重威胁.空间数据的访问控制已成为 GIS 应用迫切需要解决的热点安全问题.

由于空间数据及其访问方法自身所具有的一些特点,传统数据库的访问控制方法并不能直接应用于空间数据库的保护,需要根据空间数据的表示方法、访问模式、实时性等特点为其建立比较有针对性的访问控制模型.空间数据的一种重要表示方法是矢量表示法[1,2],即利用欧几里德(Euclid)几何中的点、线、面及其组合体来表示地理实体的空间分布;同时,地理实体也具有类型、数量、状态等各种可使用常规数据类型来表示的非空间属性信息.空间矢量表示法所特有的高精度、低存储量等优点,已使其在各种 GIS 系统中得以大规模应用.对空间矢量数据的查询访问控制,面临着以下 3 个方面的需求:

(1) 地理实体的空间分布与非空间属性涉及国家安全、科技协作交流和知识产权保护等各个方面,对其进行分级保护的需求很强烈;同时,矢量数据访问控制系统应当确保授权策略覆盖所有矢量数据.因此,与自主访问控制模型相比,强制访问控制模型更适于保护空间矢量数据库.

(2) 地理实体的空间矢量分布极不规则,常常跨越多个不同级别的敏感区域分布.如,一条公路既可以穿过居民区也可以穿越军事区,一座金矿可能跨越多个国家的边界分布.将访问控制的最小保护粒度设置在地理实体一级,将无法区别对待不同敏感区域中的实体片段,应对这些地理实体进行细粒度的拆分后加以保护.

(3) 地理实体的矢量拓扑分析过程复杂且费时,在对空间矢量数据的访问授权判定过程中对地理实体进行细粒度的拓扑裁剪次数过多,将严重影响前台系统的响应速率.若无法寻找到一个高效的实施机制,则空间矢量数据访问控制模型本身将成为需对海量空间地理数据进行处理的各种 GIS 应用的性能瓶颈,无法取得安全性与实用性上的平衡.

为了满足空间矢量数据安全访问控制面临的这 3 个方面的需求,本文提出一种细粒度空间矢量数据强制查询访问控制模型 SV_MAC(spatial vector data mandatory access control model):(1) 可根据安全标签信息来管理用户查询空间矢量数据的权限,确保授权控制范围覆盖所有矢量数据;(2) 在访问控制粒度上,SV_MAC 模型将空间对象的矢量分布拆分成多个具有不同安全级别的子块加以保护;(3) 在将保护能力细化的同时,把访问控制策略信息融入到空间 R+索引树中.提出空间数据检索与策略检索一体化的 AR+树索引结构,可在空间矢量数据查询过程中高效地实现 SV_MAC 的授权判定.即,在提供细粒度授权保护的同时保障了空间矢量数据访问的响应速度以及前台 GIS 应用的用户体验.

本文第 1 节介绍 SV_MAC 模型.第 2 节中提出将策略与数据检索相结合的 AR+树及基于 AR+树的模型高效实现.第 3 节中,基于模拟实验的结果和数据对实现结果展开分析.第 4 节介绍空间访问控制研究的相关工作.第 5 节为结束语.

1 空间矢量数据的强制查询访问控制模型 SV_MAC

依据地理要素数据模型[1],地球上的地理实体都拥有一个几何矢量分布属性 geometry,地理实体的 geometry 属性可使用点(集)、线(集)、面(集)来表示.空间矢量数据库将地理实体划分成多种类型,每一种称为一种要素模式,对应数据库中的一张要素表.要素表中的每一条元组对应一个地理实体,称为要素实例,记录了该实体的各种非空间属性信息和 geometry 属性信息.用户对指定空间范围内空间数据发起查询请求,将获得满足条件返回的要素实例叠加后的地图.每一种类型的要素对应于该地图中的一层图像.

SV_MAC 模型将空间矢量数据的保护粒度细化到每一个要素子块,为其设定强制访问控制标签加以保护.本节中,我们将对空间矢量数据库模型和 SV_MAC 模型分别进行形式化描述,并给出一个实例展示 SV_MAC 的实施效果.

1.1 空间矢量数据模型

定义 1(空间矢量数据库模型). 空间矢量数据库模型是一个七元组 $S=\langle \varepsilon,att,dom,N_r,D_r,A_r,Filter\rangle$.其中,

(1) $att=ng\_att\cup\{geometry\}$:数据库中属性字段的集合.$ng\_att$ 为非空间属性字段集合;$geometry$ 为空间几何形状属性,记录了每个要素实例在地图上的空间几何形状,可以为点、线、面.

(2) $\varepsilon=\langle E_1,E_2,...,E_m\rangle$是数据库中所有要素表模式的集合,每个空间要素表模式 E_i 是一个 n 元组$\langle e_1,\ldots,e_n\rangle$.其中,$e_1,\ldots,e_{n-1}\in ng\_att$,为非空间属性;$e_n=geometery$.

(3) dom:所有 att 中属性的属性值域并集.

(4) $N_r:\varepsilon\to N$ 将每个要素表模式关联到其所包含的属性字段的个数.

(5) $D_r:att\to 2^{dom}$ 将每个属性字段关联到其属性值域.

(6) $A_r:\varepsilon\times N\to att$ 将每个要素表模式的第 n 个属性位关联到一个空间或非空间属性.

(7) $Filter$:空间数据过滤器 $filter$ 的集合.一个空间数据过滤器 $filter$ 是一个二元组$\langle nongeo_cons,geo_win\rangle$.其中,$nongeo\_cons$ 是一个非空间属性值过滤条件集合,其中所包含的每一个非空间属性值过滤条件 $con=\langle featureType,assertion\rangle$,$featureType$ 是要素表模式标识,$assertion$ 是对 $featrueType$ 类型的要素实例在非空间属性上的若干属性值约束的逻辑表达式,$nongeo\_cons$ 中任意两个过滤条件 con 的 $featureType$ 不相同;geo_win 是一个空间矩形窗口.

定义 2(空间矢量数据库实例). 给定一个空间矢量数据库模型 S,它的一个实例 $I(S)$是一个三元组$\langle I(\varepsilon),F_t,I_r\rangle$,其中,

(1) $I(\varepsilon)=I(E_1)\cup\ldots\cup I(E_m)$是所有要素实例集合,其中每一个 $I(E_i)=\{instance_j\}$是一个符合 E_i 要素表模式的要素实例的集合.

(2) $F_t:I(\varepsilon)\to\varepsilon$将每一个要素实关联到它符合的要素表模式.

(3) $I_r\subseteq I(\varepsilon)\times S.Filter$,是要素实例匹配关系,任意 $ir=(instance,filter)\in I_r$,满足 $instance.geometry\cap filter.geo\_win\neq\varnothing$,且$\exists(ft,assertion)\in nongeo_cons$,使得 $F_t(instance)=ft$,且 $instance$ 的非空间属性值满足 $assertion$.

定义 3(空间矢量数据查询). 针对空间矢量数据库实例 $I(S)$发起的空间矢量数据查询请求 $query$ 是一个二元组$\langle sub,filter\rangle$,其中,$Sub$ 为数据访问主体,$filter\in S.Filter$ 为空间数据过滤器.$query$ 是由主体 sub 发起的对 $filter.geo\_win$ 窗口内满足 $filter.nongeo\_cons$ 中某一过滤条件的要素实例的查询请求.每一个空间矢量数据查询请求 $query$ 返回的结果为一个经过 geo_win 裁剪的要素实例集合 $I(query)=\{ins_j\}$,由所有满足以下条件的实例成员 ins_j 组成:

(1) 存在一个唯一要素实例 $instance_j\in I(\varepsilon)$,$(instance_j,query.filter)\in I(S).I_r$,$ins_j$ 和 $instance_j$ 之间满足如下的关系:$F_t(ins_j)=F_t(instance_j)$,$ins_j.e_1=instance_j.e_1,\ldots,ins_j.e_{n-1}=instance_j.e_{n-1}$,$n=N_t(E_i)$,$ins_j.geometry=filter.geo\_win\cap instance_j.geometry$.

(2) 对于任意 $ins_j,ins_i,i\neq j$,其所对应的 $instance_j,instance_i$ 互不相同.

1.2 空间矢量数据强制查询访问控制模型SV_MAC

SV_MAC 模型的主要思想是,以一种简单而灵活的方法为空间矢量数据设定强制访问控制标签,对 BLP 模型[3]中的简单安全特性进行扩展,确保任意主体 S 若能够查询读取数据库中的数据客体 O,则必须满足 $Lable(O)\preccurlyeq able(S)$.下面我们给出模型组件的形式化定义:

定义 4(空间数据安全标签设定策略). 每一条标签设定策略是一个三元组$\langle num,filter,label\rangle$,其中,$num$ 是策略编号;$filter$ 是一个空间数据过滤器;$label$ 是一个二元标签$\langle category,class\rangle$,$category$ 是范畴集,$class$ 是密级.标签设定策略说明了每一个与 $filter$ 匹配的要素子块属于 $category$ 中的每一个范畴,且密级不低于 $class$.

定义 5(添加标签的空间矢量数据库实例). 添加标签的空间矢量数据库实例是一个四元组 $I_s(S)=\langle I(S),Label,Pset,MSD\rangle$,其中,

(1) $I(S)$见定义 2.

(2) $Label$ 是所有标签的集合.

(3) $Pset\subseteq N\times Filter\times Label$ 是安全标签设定策略的集合,至少包含 1 条缺省规则$(1,filter_{all},bottemlabel)$.其中,$filter_{all}=(\{(E_1,null),\ldots,(E_m,null)\},*)$,该过滤器匹配该空间矢量数据库中所有要素实例的所有元数

据;$bottemlabel$=⟨{},$botclass$⟩,其中,$botclass$ 是最低密级,因此,$bottemlabel$ 是最低级别的标签,任意 $label\in Label$,$bottemlabel\preccurlyeq label$.

(4) MSD 是具有某一确定安全级别的要素子块集合,每一个要素子块 msd 是一个三元组⟨$instance$,$subgeometry$,$label$⟩.其中,$instance\in I(S).I(\varepsilon)$,$label$ 为该要素子块的安全级别标签,$subgeometry$ 是包含在 $instance.geometry$ 内的一个区域.MSD 集合是 $Pset$ 策略作用于 $I(\varepsilon)$上而生成的,其创建规则如下:

规则 1. 任意 $ps\in PSet$,$msd\in MSD$,若($msd.instance$,$ps.filter$)$\in I_r$,且 $msd.subgeometry\subseteq ps.filter.geo\_win$,则要素子块 msd 的 $label$ 支配该 ps 的标签,即 $ps.label\preccurlyeq msd.label$.

规则 2. 任意 $msd\in MSD$,对于其 $msd.label.category$ 中的每一个范畴 c,至少存在 1 条策略 $ps\in PSet$,满足($msd.instance$,$ps.filter$)$\in I_r$,且 $msd.subgeometry\subseteq ps.filter.geo\_win$,$c\in ps.label.category$.

规则 3. 任意 $msd\in MSD$,对于其 $msd.label.class$,至少存在 1 条策略 $ps\in PSet$,满足($msd.instance$,ps,$filter$)$\in I_r$,且 $msd.subgeometry\subseteq ps.filter.geo\_win$,$ps.class=msd.label.class$.

规则 4. 任意 $msd\in MSD$,对于任意 $subgeometry'\supset msd.subgeometry$,$msd'$=⟨$msd.instance$,$subgeometry$,$msd.label$⟩不能同时满足规则 1~规则 3.

下面我们给出在添加标签的空间数据库中 SV_MAC 模型作用下的受控查询返回结果的定义.

定义 6(受控空间矢量数据查询). 在添加标签的空间数据库实例 $I_s(S)$中,每一个空间矢量数据请求 $query$=⟨sub,$filter$⟩的返回结果为所有符合 $query$.$filter$ 且经过授权裁剪的要素实例集合 $I_s(query)=\{ins_j\}$,由所有满足如下条件的裁剪实例 ins_j 构成:

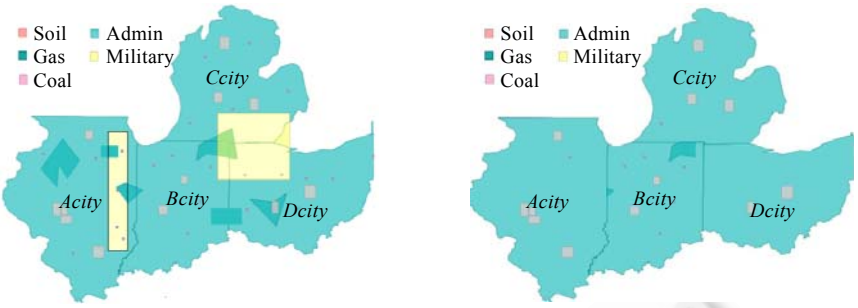
(1) 存在唯一要素实例 $instance_j\in I(\varepsilon)$,($instance_j$,$query.filter$)$\in I(S).I_r$,$ins_j$ 和 $instance_j$ 满足如下关系:$F_t(ins_j)=F_t(instance_j)$,$ins_j.e_1=instance_j.e_1$,$ins_j.e_2=instance_j.e_2$,…,$ins_j.e_{n-1}=instance_j.e_{n-1}$,$n=N_r(E_i)$,$ins_j.geometry=filte.geo\_win\cap(\cup\{subgeometry_i|\exists msd\in MSD,subgeometry_i=msd.subgeometry,msd.instance=instance_j,msd.label\preccurlyeq label(sub)\})\neq\varnothing$.

(2) 对于任意 ins_j,ins_i,$i\neq j$,其所对应的 $instance_j$,$instance_i$ 互不相同.

1.3 模型应用实例

下面我们给出一个添加标签的空间数据库的受控空间矢量数据查询实例.该实例中的空间数据库包括 4 张要素表:$admin$,$coal$,$soil$,gas,分别记录了 V 省境内的所有行政市区、煤矿、稀土矿、油田要素的相关信息.V 省境内的所有要素表信息在前端叠加显示效果如图 1(a)所示.标签集合 $Label$ 中包括的所有 $category$ 范畴为 A,B,C,D,分别对应于 A 市、B 市、C 市和 D 市的管辖范围;$class$ 集合包括了 3 个密级:$topsecret$,$secret$,$public$.$Pset$ 中包含 10 条安全标签设定策略,其中,策略 1 为缺省策略,为所有的要素表中的所有要素实例设定最低级别的标签;策略 2 将图 1(a)中军事区范围内的所有矿产资源的标签密级设定为 $topsecret$;策略 3~策略 6 将 A~D 这 4 市境内的所有石油资源的范畴设定为其所在市对应的范畴,密级设定为 $secret$;策略 7~策略 10 将 4 市境内所有储量大于某一限定值的稀土矿的范畴设定为其所在市对应的范畴,密级设定为 $secret$.

设主体 Tom 和 Jerry 同时对该数据库发起了对 V 省境内所有要素的查询请求,并已知 Jerry 为 V 省省长,主体标签为⟨{A,B,C,D},$topsecret$⟩;而 Tom 为 B 市市长,其主体标签设定为⟨{B},$secret$⟩.则对于 Jerry 和 Tom 来说,其受控空间矢量数据查询的返回结果分别如图 1(a)和图 1(b)所示.从图中可以看出,Jerry 的标签控制了所有要素表实例的安全标签,因此可以看到所有要素实例的完整显示.而对于 Tom,A 市、C 市、D 市内的资源、储量高于某一限定值的稀土矿资源以及军事区内的矿产资源均是不可见的,必须经过严格的授权裁剪.因此,最终呈现给 Tom 的空间视图是授权裁剪后的视图.



(a) Complete geo resource view in *V* province
(a) *V* 省地理资源完整视图

(b) Authorized geo resource view in *V* province
(b) *V* 省地理资源授权视图

Fig.1
图 1

2 基于 AR+树的模型高效实现

2.1 授权AR+树

R+树是一种具有高效查询性能的空间索引树[4,5],在 R+树中,每个空间实体对象与其相交的所有包围矩形相关联,且树中的所有包围矩形(除了叶子节点对应的包围矩形之外)都不重叠.这样做的结果是,当从根节点查找一个空间对象时,可以有好几条路径.这虽然增加了叶节点的冗余度,但提高了查找速度.我们对 R+树进行改进,在其叶节点和中间节点上叠加分解后的安全标签设定策略.提出一种 AR+索引树,使得基于该树的空间数据查询过程与授权检查过程可以相结合,在查找数据的同时完成授权裁剪.本节对 AR+树进行介绍.AR+树的节点结构描述如下:

- 叶子节点:($COUNT$,0,$\langle OI_1,MBR_1,PID_1\rangle$,…,$\langle OI_M,MBR_M,PID_M\rangle$,$APSET$,$CPSET$).
- 中间节点:($COUNT$,$LEVEL$,$\langle CP_1,MBR_1\rangle$,…,$\langle CP_M,MBR_M\rangle$,$APSET$,$CPSET$).

其中,$COUNT$ 为节点中的索引项或数据项个数;$LEVEL$ 指示该节点在树中的层数;0 表示叶节点;$\langle OI_i,MBR_i,PID_i\rangle$称为授权数据项,$OI_i$ 为空间要素实例的标识 ID,MBR_i 为该目标在空间中的最小包围矩形,简称为数据矩形;$\langle CP_i,MBR_i\rangle$称为索引项,$CP_i$ 为指向子树根节点的指针,MBR_i 代表其子树索引空间,为包围其子树根节点中所有目录矩形或数据矩形的最小包围矩形,简称为目录矩形.AR+树的根节点可为中间节点也可为叶子节点,其覆盖区域视为全平面(见表 1).

$APSET$,$CPSET$,PID 为与安全标签设定策略相关的数据结构.令 $Pset_i$ 为添加标签的空间数据库实例 $I_S(S)$的 $Pset$ 中所有 $filter$ 匹配要素表模式 E_i 的化简安全标签设定策略的集合,即,$Pset_i=\{rule_i|rule_i.filter.nongeo\_cons=\{(E_i,assertion)\}$,$\exists rule\in Pset$,s.t.$(E_i,assertion)\in rule.filter.nongeo\_cons$,$rule_i.filter.geo\_win=rule.filter.geo\_win$,$rule_i.label=rule.label$,$rule_i.num=rule.num\}$,则要素表 E_i 的 AR+索引树根节点的 $APSET$ 为 $Pset_i$ 中所有全平面策略(即 $rule.filter.geo\_win$=*)的集合,根节点的 $CPSET=Pset_i-APSET$;任意非根节点 $node$ 的 $APSET$ 和 $CPSET$ 则由其父节点 $CPSET$ 策略分解而得.其父节点的 $CPSET$ 中所有 $filter.geo\_win$ 覆盖了该 $node$ 目录矩形的策略 $oldrule$,分解成 $newrule.filter.geo\_win=oldrule.filter.geo\_win\cap mbr$ 的新策略 $newrule$,记录于该 $node$ 的 $APSET$;而未覆盖该 $node$ 目录矩形的策略 $oldrule$,分解成 $newrule.filter.geo\_win=oldrule.filter.geo\_win\cap mbr$ 的新策略 $newrule$,记录于该 $node$ 的 $CPSET$.每一个叶子节点的每一授权数据项$\langle OI_i,MBR_i,PID_i\rangle$中的 PID_i 为该叶子节点的 $CPSET$ 集合中与 MBR_i 相交且适用于空间要素 OI_i 的标签设定策略编号集.在图 2 中,我们给出了第 1.3 节示例中 gas 要素表的 AR+树示例.

Table 1 An example of labeled spatial database instance

表 1 添加标签的空间数据库实例示例**

Element	Content
I(*S*)	Admin table: ID / Name / Area / Geometry — 1 / *Acity* / 1 200 / *Acity area*; 2 / *Bcity* / 1 300 / *Bcity area*; 3 / *Ccity* / 790 / *Ccity area*; 4 / *Dcity* / 2 200 / *Dcity area* Gas table: ID / Name / Geometry Coal table: ID / Name / Geometry Soil table: ID / Name / Reserves / Geometry
Label	category: {*A*,*B*,*C*,*D*}; class: {*public*,*secret*,*topsecret*}
Pset	1. ⟨{(*admin*,*null*),(*gas*,*null*),(*soil*,*null*),(*coal*,*null*)},*⟩; ⟨{},*public*⟩. 2. ⟨{(*gas*,*null*),(*soil*,*null*),(*coal*,*null*)},*military area*⟩; ⟨{},*topsecret*⟩. 3. ⟨(*gas*,*null*),*Acity area*,*⟩; ⟨{*A*},*secret*⟩. 4. ⟨(*gas*,*null*),*Bcity area*,*⟩; ⟨{*B*},*secret*⟩. 5. ⟨ (*gas*,*null*),*Ccity area*,*⟩; ⟨{*C*},*secret*⟩. 6. ⟨(*gas*,*null*),*Dcity area*,*⟩; ⟨{*D*},*secret*⟩. 7. ⟨(*soil*,*reserves*>2500),*Acity area*,*⟩; ⟨{*A*},*secret*⟩. 8. ⟨(*soil*,*reserves*>1000),*Bcity area*,*⟩; ⟨{*B*},*secret*⟩. 9. ⟨(*soil*,*reserves*>2000),*Ccity area*,*⟩; ⟨{*C*},*secret*⟩. 10. ⟨(*soil*,*reserves*>1000),*Dcity area*,*⟩; ⟨{*D*},*secret*⟩.

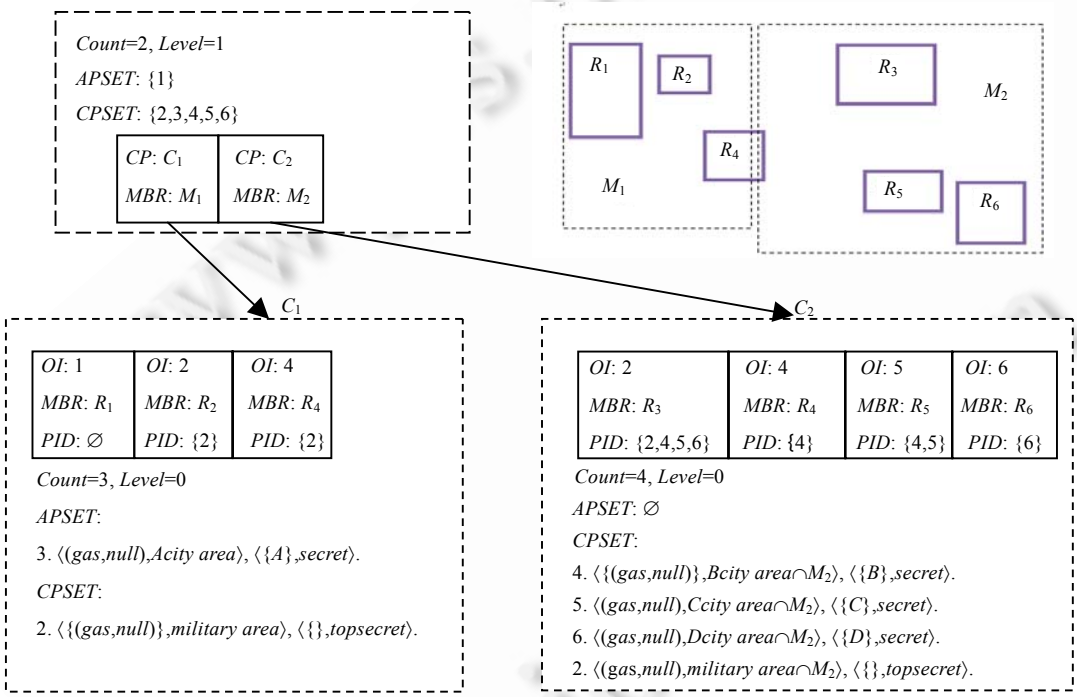


Fig.2 An example of an AR+ tree with policies in its nodes

图 2 AR+树及其各节点对应的策略示例

2.2 实 现

2.2.1 受控空间矢量数据查询

AR+树是为某一要素表构建的授权索引树,可在空间数据检索过程中同时实现授权裁剪.当查询时,受控空

** 这里省略了对 MSD 集合的描述.

间矢量数据查询 *query* 的过滤条件可能同时选择了同一空间范围内的多种要素,可分解成多个针对单一要素表 E_i 的查询 $query_i$,随后对多个要素表分别展开基于索引树的查询.我们提出算法 Arsearch,对单一空间要素表实施带授权裁剪的数据检索,其伪代码如算法 1 所示.在该算法中,空间数据的查询从 AR+树根节点开始,逐层向下搜索.下面我们直接用 *Filter* 表示 $query_i$ 中的空间过滤器 $query_i.filter$,*assertion*(*Filter*)表示 $con_i.assertion$,*assertion*(*arule*)表示检索过程中所遍历节点 *CPSET*/*APSET* 中安全标签设定策略 *arule* 的过滤器属性断言部分.

当处理 AR+树的某个节点 *T* 时,首先将查询条件 *Filter* 的 *geo_win* 设置为原始查询窗口与该节点所在 *MBR* 的交集(代码 1、代码 2).随后检查 *T* 的 *APSET* 中每一条策略,如存在某条策略 *arule* 的 *label* 无法被查询主体的 *label* 控制,则进一步比对 *assertion*(*arule*)和 *assertion*(*Filter*):如果任意满足 *assertion*(*Filter*)的要素实例都能满足 *assertion*(*arule*),则认为它们匹配.由于 *APSET* 中任意分解策略的作用窗口等于 *T* 所在的 *MBR*,因此可断定 *T* 子树中存放的节点均无法满足授权要求,可提前结束对 *T* 子树的查询,返回(代码 4、代码 5);如果 *assertion*(*Filter*)与 *assertion*(*arule*)不匹配,则修改 *assertion*(*Filter*)(代码 6).

当 *T* 为叶节点时,要检查其中的每个授权数据项,找出所有与查询区域相交且属性值符合查询条件的空间对象实例,依据 *Filter* 对其在结果集 *RESULT* 中所对应的 *Resultinstance*(*OID*)进行空间裁剪和合并(若该实例不在结果集中,需先为其创建一个 *ResultInstance*(*OID*)),并进一步将数据项 *PID* 中所有标签不受主体标签控制的策略号添加到 *Resultinstance*(*OID*).*ruleset* 中(代码 7、代码 15);当 *T* 为中间节点时,对 *T* 的每一个索引项 *E* 进行检查,如果 *E* 对应的 *MBR* 与 *Filter* 中的查询区域 *Q* 有交叠,则查询在 *E* 对应的子树递归进行,反之,则放弃 *E* 而转向 *E* 的下一个兄弟节点(代码 16、代码 20);若 *T* 为根节点,处理完 *T* 的所有索引项/数据项后,需要进一步对 *RESULT* 结果集进行处理,处理方式为,使用每一条 *Resultinstance*(*OID*)的 *ruleset* 中的匹配策略对 *Resultinstance*(*OID*)进行授权裁剪,最终返回裁剪后的 *RESULT* 集合(代码 21、代码 24).

算法 1. *search*(*S*,*R*,*Filter*).

输入:*S*:发起查询的主体;*R*:起始查询节点;*Filter*:查询过滤器.

输出:*RESULT* 结果集,为全局变量,包含所有符合查询条件,且经过授权裁剪的要素实例.

```
1.  令 W=Filter.geo_win=Filter.geo_win∩MBR(R);
2.  If (W==∅) then return;
3.  For R.APSET 中的每一条分解策略 arule
4.      if ¬(S.label≽arule.label) Then
5.          if assertion(arule)与 assertion(Filter)匹配 Then Return;
6.          Else assertion(Filter)=assertion(Filter)∧(¬assertion(arule));
7.  if R 是叶节点
8.      For R 的每一数据项(OID,MBR,PID)
9.          if Instance(OID)与 Filter 匹配 Then
10.             If ResultInstance(OID)不在 RESUL 中 Then
11.                 在 RESULT 中创建 ResultInstance(OID)
12.             ResultInstance(OID).geometry∪=Instance(OID).geometry∩Filter.geo_win;
13.             for PID 中的每一条策略 prule
14.                 If ¬(S.label≽prule.label) Then
15.                     将 prule.num 添加到 ResultInstance(OID).ruleset 中
16. Else If R 是中间节点
17.     For R 的每一索引项(CP,MBR)
18.         if MBR 与 W 相交 Then
19.             Filter.geo_win=W∩MBR;
20.             Search(S,CP,Filter);
```

21. If *R* 是根节点
22. For *RESULT* 中的每一个 *ResultInstance*(*OID*)
23. For *ResultInstance*(*OID*).*ruleset* 中的每一个 *prule*
24. *Resultinstance*(*OID*).*geometry*=*prule*.*filter*.*geo_win*

2.2.2 AR+树构建

AR+树的构建过程分两个步骤:第 1 步为要素表构建一棵 R+树;第 2 步是授权叠加,即将标签设定策略裁剪后叠加到树的每一个中间节点上.构建 R+树已经有了完备的算法,算法 2 为我们提出的授权叠加实现算法 *authoriazedARPlusTree* 的伪代码.对于一棵新建的 R+树,我们为其每一个中间节点添加空的 *APSET*,*CPSET* 集合,生成一棵初始 AR+树.*authorizedAPLusTree* 从根节点开始对该 AR+树进行授权叠加:首先,将 *Pset* 中的所有全平面规则和所有非全平面规则分别添加到 *Root* 的 *APSET*,*CPSET* 中;随后,检查 *Root* 的每一个索引项/数据项,调用 *Decompose* 算法(见算法 3),递归地将 *CPSET* 中的安全标签设定策略分解叠加到索引项/数据项中.

算法 2. *AuthorizedRPlusTree*(*RTree*,*Pset*).

输入:*RTree*:*Apset*,*Cpset* 均为空的 AR+树;*Pset*:标签设定策略集.

输出: *ARTree*:将 *Pset* 安全标签设定策略叠加到 *Rtree* 上后的索引 AR+树.

1. *Root*=*Rtree*.*root*;
2. For *Pset* 中的每条策略 *rule*
3. If *rule*.*filter*.*geo_win*==*WholeArea* Then
4. 将 *rule* 添加到 *Root*.*APSET*;
5. Else 将 *rule* 添加到 *Root*.*CPSET*;
6. For *Root* 中的每一个索引项/数据项 *term*
7. *Decompose*(*term*,*Root*.*CPSET*);

算法 3. *Decompose*(*term*,*ParentCPSET*).

输入:*term*:索引项/数据项;*ParentCPSET*:该索引项/数据项所在节点的 *CPSET*.

输出:无.

1. If ((*term*.*CP*)→*child*==*null*) Then //叶子节点
2. For *ParentCPSET* 中的每条策略 *rule*
3. If *rule*.*filter*.*geo_win*∩*term*.*MBR*≠∅ then
4. 将其初始策略号添加到 *term*.*PID* 中
5. Else
6. *Currentnode*=(*term*.*CP*)→*node*.
7. For *ParentCPSET* 中的每条策略 *rule*
8. if *rule*.*fitler*.*geo_win* 覆盖了 *term*.*MBR* Then
9. 将 *Rule* 添加到 *Currentnode*.*APSET*
10. else If *rule*.*filter*.*geo_win*∩*term*.*MBR*≠∅ Then
11. *Newrule*=*rule*;
12. *Newrule*.*filter*.*geo_win*=*rule*.*filter*.*geo_win*∩*MBR*;
13. 将 *newrule* 添加到 *Currentnode*.*CPSET*;
14. For *Currentnode* 中的每一个 $term_i$
15. *Decompose*(*termi*,*Currentnode*.*CPSET*);

2.2.3 标签设定策略插入和删除

安全需求的更改会造成空间数据库内的标签设定策略发生变动,从而引起 AR+树的变动.当一条新的标签设定策略 *rulenew* 插入 *Pset* 时,只需在 AR+索引树的树根上执行一遍 *authorizedRPlusTree*(*oriARTree*,*rulenew*)

即可.而当一条已有的标签设定策略 *rule* 要被删除时,也只需将所有节点的 *APSET*,*CPSET* 以及所有数据项 *PID* 中策略号等于 *rule.num* 的分解策略删除即可.

2.2.4 要素实例数据插入和删除

在空间数据库中插入和删除空间要素实例也会引起 AR+树自身的变更,其过程与 R+树不同的是:(1) 在 AR+树中,当任一中间节点 *node* 中的授权索引项(CP_i,MBR_i)的目录矩形 MBR_i 扩展时,CP_i 所指向的节点 *cnode* 的 *APSET*,*CPSET* 将被重新计算,计算结果为 *node.CPSET* 中的每一条 *rule* 与扩展后的新的 MBR_i 交叠裁剪形成的新 *rule* 的集合,而 *node* 的子节点并不受影响,除非其子节点的授权索引项的目录矩形也需扩展.(2) 当任一节点 *node* 要分裂成两个节点 $node_1$ 和 $node_2$ 时,原先的节点 *node* 及其子树节点自然销毁,分裂后新生成的两个以 $node_1$ 和 $node_2$ 为根的子树 $tree_1$ 和 $tree_2$,它们最初的状态是所有节点 *APSET*/*CPSET* 均为空的 AR+树,需对它们分别执行 *authorizedRPlusTree*($tree_i$,*parentCPSET*)算法来添加标签设定策略集.其中,*parentCPSET* 是 *node* 的父节点的 *CPSET* 集.

当有空间要素实例从表中删除时,该实例的数据矩形可能从多个叶节点中被删除,并可能引起叶节点以及中间节点的目录矩形缩减,其缩减方式同 R+树.即,调整每个节点在其父节点中的索引项(CP_i,MBR_i)的 MBR_i,同时还要重新计算 CP_i 节点的 *APSET*,*CPSET* 集合.

3 实现效率分析

本文使用对比的方法对 AR+树查询算法的性能进行模拟测试,实验使用机器的处理器主频 3.40 GHz,内存 4GB,模拟程序使用 Java 语言编写,运行于 JRE 1.6 上.在实验中,我们一方面测试了叠加了不同数量安全策略的 AR+树索引相对原始 R+树索引查询性能的下降程度,另一方面比较了 AR+树索引与传统的策略与数据分离的双 R+树索引[***]在实施 SV_MAC 模型效率上的优劣.实验主要步骤如下:

1. 生成数据集.首先生成全平面空间[*x*l=0,*y*l=0,*xh*=100000,*yh*=100000](单位:米)上的 5 个随机要素表 $Test_1$~$Test_5$,分别拥有 2 000,4 000,6 000,8 000,10 000 条空间矢量要素数据,随后生成 3 个随机安全策略数据集,分别包含 500,1 000,2 000 条安全策略.
2. 创建索引树.为每一个空间数据集创建一个 R+树索引和 3 个 AR+树索引,分别对应于 3 个安全策略集;此外,为每一个安全策略数据集创建一个 R+树索引.
3. 查询测试.为每一个空间数据集生成 2 个随机空间查询请求集合,各自包含 5 000 条查询请求,第 1 个集合中所有查询请求的查询面积为 0~4(亿米 ^{2}),第 2 个集合中的查询面积为 4~25(亿米 ^{2}).对每一个空间数据集分别使用 2 个空间查询请求集,执行基于 R+树索引的查询,基于 AR+树(500/1 000/2 000 条策略)索引的查询,以及基于双 R+树(500/1 000/2 000 条策略)索引的查询,记录查询时间.

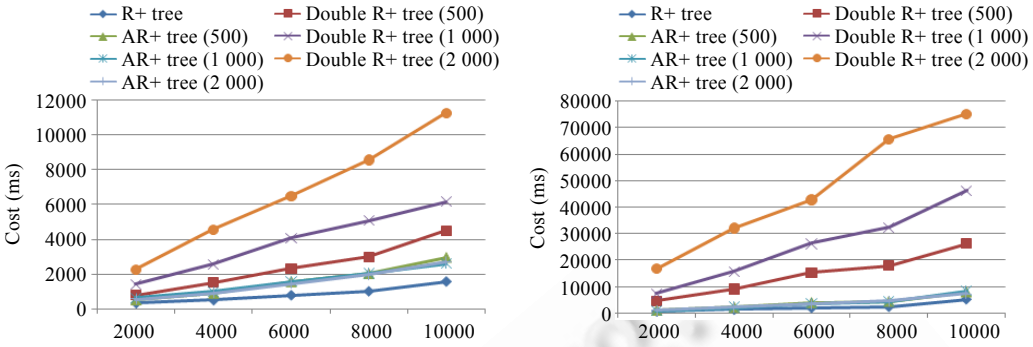
从原理上说,AR+树和其他空间索引树一样,以 geometry 属性为索引属性,从而提高了对某一空间范围内的空间要素实例查询的效率,而不影响非空间属性值(如“稀土储量”大于 3 000 万吨等)的查询效率.因此,我们在测试中生成的随机空间查询请求的非空间查询条件均为空,以集中比较 R+树、AR+树以及双 AR+树模式的空间查询效率.即,每一条查询请求形如 *query*=⟨*Sub*,((*Test*$_i$,*null*),*geo_win*)⟩.实验结果如图 3 所示.

如图 3 所示,实施强制访问控制造成的查询时间增长是不可避免的,采用双 R+树实施方法造成的增长率是不可接受的;且随着策略数和查询面积的增长而大幅增长,使用 2 000 条安全策略对 6 类空间数据集进行查询测试,0~4(亿米 ^{2})内的查询时间平均增长了 7.9 倍,4~25(亿米 ^{2})内的查询时间平均增长了 21 倍;而 AR+树的实施方法带来的增长率则显著降低,其中,使用 2 000 条安全策略对 6 类空间数据集进行查询测试,0~4(亿米 ^{2})内的查询时间平均只增长了 1.8 倍,而 4~25(亿米 ^{2})内的查询时间平均增长了 1.6 倍.

此外,从图 3 中可以看出,AR+树的实施方法带来的增长率并不会随着策略数的增长而显著增长.这是因为

*** 即为策略和空间数据分别建立 R+树,在接收到空间数据请求时,对两棵 R+树分别执行空间查询得到实例查询结果集 *A*1 和策略查询结果集 *A*2,随后使用 *A*2 中的策略对 *A*1 中的每一条数据进行授权裁剪.

AR+树不仅将策略叠加到了叶子节点,而且将相当一部分数量的策略叠加到了中间节点.当策略数增加时,一方面,其叶子节点的数据项中存储的策略数的增加幅度并不大;另一方面,中间节点中存储的策略数增加使得中途查询返回的概率增大,节省了大量对非授权子树的查询时间和策略比较时间.



(a) In about 0~0.4 (billion m^2)
(a) 0~4(亿米 ^{2})内

(b) In about 0.4~2.5 (billion m^2)
(b) 4~25(亿米 ^{2})内

Fig.3 Comparison of three methods for querying features located

图 3 3 种查询模式性能比较

综上所述,基于 AR+树的 SV_MAC 模型实施将安全标签设定策略检索与数据检索融合起来,大大降低了访问控制对实际数据检索带来的效率损耗,与传统的双 R+树检索方式相比具有非常明显的优势,是一种高效的 SV_MAC 模型实施方法.

4 相关工作

空间数据访问控制模型的研究工作已得到许多学者的关注,目前的研究思路主要是对传统访问控制模型的授权方法进行空间扩展.在这方面的工作中做出主要贡献的学者是 Chun 和 Atluri 等人.在文献[6]中,Chun 和 Atluri 等人对传统的自主访问控制进行了空间扩展,提出了针对栅格数据的空间授权模型,并给出了系统的实现框架.Atluri 等人在文献[7]中提出一种可对指定空间区域和数据时间的栅格数据和矢量数据进行授权的方法 GSAM.在 GSAM 中,一个授权 *a* 被指定为 4 元组$\langle ce,ge,pr,\gamma\rangle$.其中,*ce* 是一个表示被授权的主体的证书表达;*ge* 是对要授权的时空对象的描述;*pr* 是一系列特权模式,表示被允许的操作;γ为时间段,表示在这段时间内访问是被允许的.Belusi 等人在文献[8]中提出了一种针对 Web 地图服务中矢量数据的授权模型.该模型可表示为$\langle r,fc,p,w,gr,gr\ op\rangle$,其分量分别表示授权角色、要素类、特权行为、授权窗口、授权者和授权结果.

然而,上述模型均扩展于传统的自主访问控制模型,易于绕过,在保护力度上弱于强制访问控制模型,难以有力地保护一些具有军事、经济、政治敏感性的空间地理数据;另一方面,模型中未考虑到矢量数据的细粒度授权保护问题,难以精确地保障目前大规模应用的空间矢量数据的数据库安全.

学者们也对空间访问控制模型的高效策略存储和检索实现方法展开了研究.这方面的研究工作主要是在 Atluri 等人提出的 GSAM 授权模型基础上开展的,其研究思路是,利用空间数据常用的索引结构,在其中加入授权信息,加速授权策略的检索效率,降低授权管理代价.文献[9]中提出针对空间栅格数据进行区域划分,基于四叉树结构建立索引,并在相应区域上添加授权信息,使得在对空间数据查找过程中能够快速判断其操作的有效性.文献[10]将文献[9]的工作扩展到 R 树结构上,去掉对空间授权区域是正方形的限制,允许空间区域有交集,而且支持多分辨率图像授权索引机制.文献[11]则将带有授权的索引结构扩展到移动数据库基础上,可对用户基于时间变化的数据进行有效的判定.

其中,文献[10]的工作与我们最为相关,因其支持矢量数据的访问控制策略检索,且其所基于的 R 树索引是我们所选用的 R+树索引的原型.文献[10]的工作虽然支持矢量数据的访问控制实现,但却不适于支持对矢量数

据的细粒度拆分保护.一方面因为其所考虑的访问控制模型本身未提供此项支持,另一方面是因为 R 树索引中间目录矩形的可重叠性导致其查询效率低于 R+树(R 树目录矩形会存在重叠,导致一个空间实体在空间上可能包含于多个节点,但在存储时它只与 1 个节点相关联.这就意味着一个空间查询常常需要访问好几个节点以最终确定一个对象是否存在,因此在查询处理时会带来额外的开销,使其查询效率低下),对其叠加细粒度的矢量数据授权策略后,授权策略被冗余地叠加于各个重叠的中间节点之上,在查询时进一步增加了额外的非必要授权裁减开销,严重影响前台系统的响应速率.

5 结束语

本文提出一种细粒度的空间矢量数据强制查询访问控制模型 SV_MAC 以及一种将空间数据与安全策略检索相结合的 AR+树索引结构,以在空间数据查询过程中高效实现 SV_MAC 模型.实验结果表明,该方案在为空间矢量数据检索提供细粒度强制安全保护的同时,保障了前台响应速度.下一步的工作中,我们将对空间矢量数据的添加、删除、修改等写操作展开研究,设计适于其所面向的专业测绘用户群的访问控制模型.

References:

[1] OGC reference model (ORM). 2003. http://www.opengeospatial.org/standards/orm#_Toc87953568

[2] Cui TJ. Principles of Geospatial Databases. Beijing: Science Press, 2007 (in Chinese).

[3] Bell DE, LaPadula LJ. Secure computer system: Unified exposition and multics interpretation. Technical Report, MTR-2997 Rev.1, Bedford: The MITRE Corporation, 1976.

[4] Zhang MB, Lu F, Shen PW, Cheng CX. The evolvement and progress of R-tree family. Chinese Journal of Computers, 2005,28(3): 289−300 (in Chinese with English abstract).

[5] Guo W, Guo J, Hu ZY. Spatial Database Indexing Techniques. Shanghai: Shanghai Jiaotong University Press, 2006 (in Chinese).

[6] Chun SA, Atluri V. Protecting privacy from continuous high-resolution satellite surveillance. In: Thuraisingham BM, van de Riet RP, Dittrich KR, Tari Z, eds. Proc. of the IFIP TC11/WG11.3 14th Annual Working Conf. on Data and Applications Security. Catalonia: Springer-Verlag, 2000. 233−244. [doi: 10.1007/0-306-47008-X_21]

[7] Atluri V, Chun SA. An authorization model for geospatial data. IEEE Trans. on Dependable and Secure Computing, 2004,4(1): 238−254. [doi: 10.1109/TDSC.2004.32]

[8] Belussi A, Bertino E, Catania B, Damiani ML, Nucita A. An authorization model for geographical maps. In: Cruz IF, Pfoser D, eds. Proc. of the 12th ACM Int'l Workshop on Geographic Information Systems (ACM-GIS). Washington: ACM Press, 2004. 82−91. [doi: 10.1145/1032222.1032236]

[9] Atluri V, Mazzoleni P. A uniform indexing scheme for geo-spatial data and authorizations. In: Gudes E, Shenoi S, eds. Proc. of the IFIP TC11/WG11.3 16th Conf. on Data and Application Security. Catalonia: Springer-Verlag, 2003. 207−218.

[10] Atluri V, Guo Q. STAR-TREE: An index structure for efficient evaluation of spatiotemporal authorizations. In: Farkas C, Samarati P, eds. Proc. of the IFIP TC11/WG 11.3 18th Annual Conf. on Data and Applications Security. Catalonia: Springer-Verlag, 2004. 31−47. [doi: 10.1007/1-4020-8128-6_3]

[11] Atluri V, Shin H, Vaidya J. Efficient security policy enforcement for the mobile environment. Journal of Computer Security, 2008, 16(4):439−475.

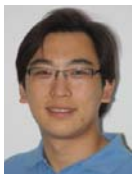
附中文参考文献:

[2] 崔铁军.地理空间数据库原理.北京:科学出版社,2007.

[4] 张明波,陆峰,申排伟,程昌秀.R 树家族的演变和发展.计算机学报,2005,28(3):289−300.

[5] 郭薇,郭菁,胡志勇.空间数据库索引技术.上海:上海交通大学出版社,2006.

张妍(1983－),女,福建邵武人,博士,主要研究领域为分布式授权,空间数据库安全.



陈驰(1978－),男,博士,高级工程师,主要研究领域为信息系统安全.

冯登国(1965－),男,博士,研究员,博士生导师,主要研究领域为网络与信息系统安全,密码理论与技术.