强制存取控制策略在数据库安全中的应用

林大松 （北京国安电气总公司 100080）
俞 捷 （首都经贸大学信息系 100026）

摘要：本文提出了一种在多层次关系数据模型中采用强制存取控制策略实现对信息分级分类进行管理，强行限制系统中信息的流动和共享，以保护信息安全的方法。

关键词：多层次关系数据模型 强制存取控制 多示例 存取级别

一、引言

本文着重讨论有关数据库中的存取控制问题，数据库的存取控制模型分为以下几个方面：

(1)自主访问控制：是基于一种访问控制规则实现主体对客体的访问，这种控制是自主的，自主是指某一主体能直接或间接地将访问权或访问权的某些子集授予其他主体。

(2)强制存取控制：强制存取控制是基于信息分类的，用于保护数据、防止被盗。

(3)高级 DBMS 的授权模型，如面向对象的 DBMS，在高级 DBMS 中表达的数据模型要比关系模型丰富得多，高级数据模型包括如下概念：继承结构、对象组合、版本、方法等。因此，关系数据库的授权模型必须扩展到这些增加的概念上。

二、强制存取控制策略保护数据

1. 强制存取控制策略

强制访问控制是将信息分级分类进行管理，强行限制系统中信息的流动和共享。数据库中的某些数据是敏感的，并存在不同的敏感程度，强制访问控制则限制用户存取他们仅能合法存取的数据，同时保证敏感数据不泄露给非授权用户。强制访问控制不可避免地要对用户自己的客体施加一些严格的限制。客体是存储信息的被动实体，如关系里的元组，元组里的元素，主体是能够存取客体的主动实体，通常是根据用户的利益主动操作的过程，主体和客体都要指定相应的存取级别来标识数据的不同敏感度，存取级别通常由两个元素决定：安全层和分类集合，安全层是一个层次有序集合，反映了信息的敏感度，它包括：绝密(TS)、机密(C)、秘密(S)、一般(U)，TS>C>S>U；分类集合是一个无序集合，用以说明对客体施行的存取限制。并且仅当存取级别 C1 安全层大于等于 C2 安全层，并且 C1 的分类包含 C2 的分类时，则称存取级别 C1 优于(>=)存取级别 C2，如果既没有 C1>=C2，也没有 C2>=C1，则 C1，C2 称为不可比的。

强制存取控制策略是基于以下两个原理：

(1)仅当主体存取级别优于(>=)客体存取级别时，主体对客体具有读权限；

(2)仅当客体存取级别优于(>=)主体存取级别时，主体对客体具有写权限。

在这种访问控制中，所有的访问均是根据存取级别的比较来决定是否允许访问。

2. 多层次关系数据模型

在标准的关系模型中，每个关系是由以下两个元件表示的：

(1)状态不变量关系模式 R(A1，...，An)，每个 Ai 是在某个域 Di 上的属性。

(2)状态变量关系实例集由形如(a1，...，an)的不同元组组成，ai 是域 Di 的值。

假设 R(U)是属性集 U 上的一个关系模式，X、Y 为 U 的子集，如果对于 R(U)的任意一个可能的关系 R，R 中不可能有两个元组在 X 中的属性值相同，而在 Y 中的属性值不同，则称 Y 函数依赖于 X。

关系的关键字是属性的最小集合，其他属性都是函数依赖这个集合的。主关键字唯一标识了关系中每个元组。一个关系不能包含有相同主关键值的两个元组，同时，主关键字必须有值且不能为空，这些限制称为实体完整性。

在关系数据库上应用强制存取策略需要将数据进行分类，分类工作可通过对每一个元组(行)、每一个属性

(列)、每个元素(属性值)分类来进行。

将存取级别应用到数据库上,就产生了多层关系数据库,一个多层关系可由如下两个元件表示:

(1)状态不变量多层关系模式 R(A1, C1, . . . , An, Cn, TC), TC, 每个 Ai 是某个域 Di 上的属性, Ci 是属性 Ai 的分类属性,它的域是与 Ai 的值相关的存取级别的集合, TC 是元组的分类属性。

(2) 状态变量关系实例集 Rc, 每个实例 Rc 是由形如 $(a_1, c_1, \ldots a_n, c_n, t_c)$ 的不同元组组成, a_i 是域 D_i 值, c_i 是 C_i 域中的值, t_c 是元组中所有 c_i 的最小上限的存取级别。

在分类 c_i 的关系实例中包含的所有数据对于层次 c_i 上的用户来说是可视的。根据读取等级限制的原理,这些数据的存取级别由 c_i 控制。因此,在多层次关系中,每个属性值 a_i 仅仅是对于大于等于 c_i 层上面的存取级别来说是可视的。层次 c_i 的关系实例是通过屏蔽存取级别比 c_i 高或不可比层的所有元素来获得的,其值用空值来代替。

Name	C_{name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Bob	Low	Dept1	Low	100	Low	Low
Ann	High	Dept2	High	200	High	High
Tom	Low	Dept1	Low	150	High	High

图 1

如图 1,关系有 3 个属性:姓名(Name)、系别(Dept)和工资(salary),C 代表不同属性的类别,TC 为元组的分类属性,数据存放在 High 和 Low(High > Low)两个安全层,图 2 给出了两个安全层的实例,在关系的低层实例中,高层值 Tom 的工资被低层的空值代替,因而,低层用户的空值掩盖了高层或不可比层上的值,低层使用者不能确定一个空值是实际上空还是高层的隐含值。

Name	C_{name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Bob	Low	Dept1	Low	100	Low	Low
Tom	Low	Dept1	Low	null	Low	Low

Low 实例

Name	C_{name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Bob	Low	Dept1	Low	100	Low	Low
Ann	High	Dept2	High	200	High	High
Tom	Low	Dept1	Low	150	High	High

High 实例

图 2

多层关系必须满足如下限制:

(1)对多层关系的每个元组,主关键字属性必须具有相同的存取级别。

(2)对多层关系的每个元组,非关键字的存取级别必须优于主关键字的存取级别。

以上限制保证了多层关系的每个实例都满足关系数据库的完整性,如果不满足上述限制,关系实例中的某些非关键字属性将以非空值存在,而某些关键属性将以空值存在,这就破坏了实体的完整性。

3. 多示例

在标准关系模型中,每一个元组都是由关键字属性唯一标识的,在多层关系中,当考虑安全级别的问题时,可以存在多个具有相同值的元组,但其分类不同,我们把它称为多示例(指一个记录可以出现许多次,每次都具有不同的保密层次)。多示例与关系、元组以及元素的关系如下:

(1)多示例关系是一种关系,该关系是由相同的关系名字,而模式由不同的存取级别来标识的。

(2)多示例元组,也称实体多示例,是由相同的主关键字但具有不同的存取级别的元组组成,如图 3。

Name	C_{name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Bob	Low	Dept1	Low	100	Low	Low
Ann	High	Dept2	High	200	High	High
Tom	Low	Dept1	Low	150	High	High
Ann	Low	Dept1	Low	100	Low	Low

图 3

(3)多示例元素,也称属性多示例,它是由主关键字及其存取级别相同,但某些属性的值及其存取级别不同的元组组成,如图 4。

Name	C_{name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Bob	Low	Dept1	Low	100	Low	Low
Ann	High	Dept2	High	200	High	High
Tom	Low	Dept1	Low	150	High	High
Tom	Low	Dept1	Low	100	Low	Low

图 4

满足如上条件可以防止一个敏感的客体信息直接传播到低层次或不合适层次上。多示例应用有两种情况:

·当一个低层次用户在一个较高的或不可比层次的数据域中插入/更新数据时(不可视的多示例);

·当一个较高层次的用户在一个低层次的数据域中

插入/更新数据时(可视的多示例)。

①不可视多示例。假设一个低级用户要求用与高层或不可比层一样的主关键字插入一个元组,DBMS 有三种可能的选择:

a.通知用户具有相同的主关键字的元组已经存在并拒绝插入;

b.用低层上的新元组替换高层或不可比层上的元组;

c.在低层插入新元组而不修改高层或不可比层上的元组(即多示例实体);

选择 a 是不恰当的,因为它意味着在低层次上的用户已经知道高层上的信息,违反了低层不能读取高层上的信息的原则;选择 b 也是不恰当的,因为它使得低级用户能够重写对他不可见的数据,因而,违反了完整性限制;选择 c 是唯一合理的考虑。

例如,考虑图 1 中的关系数据库,假设低层用户做如下操作:

```
INSERT INTO EMPLOYEE
  Values(Ann, Dept1, 100)
```

关系中已有主关键字为"Ann"、主关键字存取级别为 High 的元组,它对低层用户是不可视的,因而直接插入新元组而不修改原有的元组。图 3 中具有主关键字"Ann"的元组是多示例。

又如,低层用户想进行更新操作:

```
UPDATE EMPLOYEE
  SET Salary = '100'
WHERE Name = 'Tom'
```

虽然"Tom"的工资域已有值,但该值的存取级别为 high,因而对低层用户不可视,所以,低层用户插入新值而不修改原有的元组,如图 4。

②可视多示例。假设一个高级用户要求用与较低层一样的主关键字插入一个元组,DBMS 有三种可能的选择:

a.通知用户具有相同的主关键字的元组已经存在并拒绝插入;

b.用高层上的新元组替换较低层上的元组;

c.在高层插入新元组而不修改低层上的元组(多示例实体);

对于选择 a 来说,由于高层次上的用户可以看见低层次上的元组,因此,用户可以看到冲突的存在,然而,对高层用户拒绝插入,隐含着拒绝服务的问题,因此这一选择是不合适的;选择 b 意味着删除低层上的元组,新的元组对于低层次用户来说是不可视的,因而也是不合适的;选择 c 是可行的。

以上的讨论同样可以用到高层用户更新低层次数据的数据域问题。

例如,考虑图 5 中的多层关系,假设高层用户做如下操作:

```
INSERT INTO EMPLOYEE
  Values(Ann, Dept2, 200)
```

Name	C_{name}	Dept	C_{Dept}	Salary	C_{Salary}	TC
Bob	Low	Dept1	Low	100	Low	Low
Ann	Low	Dept1	Low	100	Low	Low
Tom	Low	Dept1	Low	100	Low	Low

图 5

关系中已有主关键字为"Ann"、主关键字存取级别为 Low 的元组,因而上述的插入操作导致了"Ann"的元组是多示例,其结果如图 3。

元素多示例可类似产生,如高层用户要将图 5 中"Tom"的工资改为 150,结果如图 4。

支持多示例的模型需要考虑与多示例元组和元素有关的语义。

多示例元组是指不同的现实世界实体,例如,图 3 中属性 Name 为"Ann"的两个元组指两个不同的雇员:一个在"Dept1"工作,工资为 100,另一个在"Dept2"工作,工资为 200。

多示例元素是指同一个现实世界实体。例如,图 4 中属性 Name 为"Tom"的两个元组指在"Dept1"工作的同一个雇员,然而"Tom"的工资有两个不同的值,一个是存取级别为 Low 的工资(100),另一个是存取级别为 High 的工资(200)。

允许使用多示例的模型有 Seaview 模型、Jajodia & Sandhu 模型,这两个模型是基于多层关系并支持元组和元素多示例,在这些模型中,定义多层关系的关键字是原关键字属性,它们的分类以及关系中所有其他属性分类的结合。在 Seaview 模型中,更新/插入操作可以在关系中以指数级产生相当多非关键字属性元组,Jajodia & Sandhu 修改了 Seaview 模型,控制了由于更新所产生的元组的激增问题。

(来稿时间:1997 年 1 月)