Proc. of 16th NIST-NCSC National Computer Security Conference, Baltimore, MD, September 20-23, 1993, pages 39-52.

REFERENTIAL INTEGRITY IN MULTILEVEL SECURE DATABASES

Ravi S. Sandhu and Sushil Jajodia[1]

Center for Secure Information Systems
&
Department of Information and Software Systems Engineering
George Mason University, Fairfax, VA 22030-4444

Abstract This paper studies referential integrity in multilevel relations with element-level labeling. Our principal contribution is resolution of an impasse left by previous work in this area. We show that the previous work leaves us with a choice of either accepting referential ambiguity, or severely curtailing the modeling power of multilevel relations. We then show how to escape this impasse by eliminating entity polyinstantiation, while retaining element polyinstantiation (as an option). We also discuss how entity polyinstantiation can be securely eliminated.

Keywords: multilevel secure databases, referential integrity, polyinstantiation

1 INTRODUCTION

Referential integrity is an important component of the classical relational data model [4]. It is concerned with references from one relation to another. The principal motivation for referential integrity is to prevent dangling references across relations, such as when an employee is assigned to a non-existent department. Consideration of referential integrity in multilevel relations leads to the realization that it can result in signaling channels for leakage of secret information [3, 6, 7]. A multilevel secure relational model must cope with the possibility of these channels.

The central point of this paper is that prior work on referential integrity has left us with a choice of two undesirable alternatives. We either have referential ambiguity, which results in confusion about the meaning of data in relations; or we have serious limitations on the expressive power of multilevel relations, such as the inability to classify a relationship between unclassified entities.

Our principal contribution in this paper is to show how this unacceptable impasse can be resolved by building upon the distinction between entity and element polyinstantiation. We argue that entity polyinstantiation is so contrary to referential integrity that it must be eliminated. We also demonstrate how entity polyinstantiation can be easily prevented, by means of the usual integrity constraints in Database Management Systems. On the other hand element polyinstantiation can be tolerated if it is required for purpose of cover stories, or some similar reason. In other words, element polyinstantiation can be available as an option as needed; whereas entity polyinstantiation should be eliminated in the data model. (Note that element polyinstantiation can be securely prevented using the technique of [20], if it is not needed in a particular application.)

The paper is organized as follows. Section 2 defines a model for multilevel relations with element level labeling. In this section only individual relations are considered. Section 3 discusses

[1]This work was partially supported by the U.S. Air Force, Rome Laboratory under the contract # F30602-92-C-0002. We are indebted to Joe Giordano for his support and encouragement which made this work possible.

the semantics of polyinstantiation, including the important distinction between entity and element polyinstantiation. Some of the more subtle aspects of the definitions of section 2 are also discussed. Section 4 reviews prior work on referential integrity in multilevel relations, which leaves us in the impasse mentioned above. Section 5 describes how to resolve this impasse by eliminating entity polyinstantiation. Section 6 concludes the paper.

2 MULTILEVEL RELATIONAL MODEL

In this section, we give the basic definitions and assumptions used with multilevel relations. Our initial focus is on individual relations considered in isolation. Consideration of referential integrity, which involves two relations, is deferred until sections 4 and 5. The definitions and properties for multilevel relations given here are conceptually simpler, and different in important ways, as compared to previous work on element-level labeling [6, 11, 12, 13, 15, 16, 17, 19, 20]. The most significant difference is the requirement that there can be at most one tuple in each access class for a given entity. This gives us the simplicity of tuple-level labeling, combined with the flexibility of element-level labeling. There are also some other subtle differences in the precise formulation of various properties.

The reader is assumed to be familiar with basic concepts of relational database theory. In analogy to the usual definition of a relation, a *multilevel relation* consists of the following two parts.

Definition 1 [RELATION SCHEME] A state-invariant multilevel relation scheme which is denoted by $R(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC)$, where each A_i is a *data attribute*[2] over domain D_i, each C_i is a *classification attribute* for A_i, and TC is the *tuple-class* attribute. The domain of C_i is specified by a range $[L_i, H_i]$, $H_i \geq L_i$, which defines a sub-lattice of access classes ranging from L_i up to H_i. □

Definition 2 [RELATION INSTANCES] A collection of state-dependent *relation instances*, each of which is denoted by $R_c(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC)$; one for each access class c in the given lattice. Each instance is a set of distinct tuples of the form $(a_1, c_1, a_2, c_2, \ldots, a_n, c_n, tc)$ where each $a_i \in D_i$ and $c_i \in [L_i, H_i]$, or $a_i =$ null and $c_i \leq H_i$; and $tc \geq \text{lub}\{c_i : i = 1 \ldots n\}$.[3] Note that c_i must be defined even if a_i is null, i.e., a classification attribute cannot be null. □

We assume that there is a user-specified *apparent primary key* AK consisting of a subset of the data attributes A_i. In general AK will consist of multiple attributes. We also assume that the relation scheme is itself unclassified (or, more generally, classified at the greatest lower bound of L_i, $i = 1 \ldots n$). A tuple whose tuple class is c is said to be a c tuple. (Similarly, a subject whose clearance is c is said to be a c subject.)

We now list four integrity requirements which we feel must be satisfied by all multilevel relations. We call these the *core integrity properties*. We use the notation $t[A_i]$ to mean the value corresponding to the attribute A_i in tuple t, and similarly for $t[C_i]$ and $t[TC]$.

[2]In many cases it is useful to have an A_i represent a collection of *uniformly classified* data attributes. This extension requires straightforward modifications to our statements in this paper, which are all formulated in terms of the A_i's being individual data attributes.

[3]Note that in previous work [6, 11, 12, 13, 15, 16, 17, 19, 20] it has generally been required that $tc = \text{lub}\{c_i : i = 1 \ldots n\}$. The main reason for relaxing this requirement to $tc \geq \text{lub}\{c_i : i = 1 \ldots n\}$ is to allow a c-subject to specify the classification of individual attributes in a c-tuple. For example, let M_1 and M_2 be incomparable labels whose least upper bound is S and greatest lower bound is U. We should have some means of allowing a S-subject to instantiate a S tuple whose individual classification attributes are at, say, U, M_1, and M_2. Careful consideration of the update semantics in such situations, leads to the conclusion that a S-subject should be able to instantiate a S tuple, even if the least upper bound of the individual classification attributes turns out to be less than S.

Property 1 [Entity Integrity] Let AK be the apparent primary key of R. A multilevel relation R satisfies entity integrity if and only if for all instances R_c and $t \in R_c$

1. $A_i \in AK \Rightarrow t[A_i] \neq$ null,
2. $A_i, A_j \in AK \Rightarrow t[C_i] = t[C_j]$ (i.e., AK is uniformly classified), and
3. $A_i \notin AK \Rightarrow t[C_i] \geq t[C_{AK}]$ (where C_{AK} is defined to be the classification of the apparent primary key). □

The first requirement is exactly the definition of entity integrity from the standard relational model, and ensures that no tuple in R_c has a null value for any attribute in AK. The second requirement says that all attributes in AK have the same classification in a tuple. This will ensure that AK is either entirely visible, or entirely null at a specific access class c. The final requirement states that in any tuple the class of the non-AK attributes must dominate C_{AK}. This rules out the possibility of associating non-null attributes with a null primary key. Property 1 is identical to the entity integrity property of SeaView [17].

Notation. In order to simplify our notation, we will henceforth use A_1 as synonymous to AK, i.e., A_1 and AK both denote the apparent primary key.

The next property is concerned with consistency between relation instances at different access classes. It requires that at every access class c, exactly those tuples whose access class is dominated by c are visible.

Property 2 [Inter-Instance Integrity] A multilevel relation R satisfies the inter-instance integrity property if and only if for all $c' \leq c$ we have $R_{c'} = \{t \in R_c \mid t[TC] \leq c'\}$. □

Thus, for example, a TS-subject will see the entire relation given in figure 1, while a C-subject will see the filtered instance given in figure 2. Let us denote the relation between $R_{c'}$ and R_c described in property 2 by $R_{c'} = \sigma(R_c, c')$, where σ is called the *filter function.* It is evident that $\sigma(R_c, c) = R_c$, and $\sigma(\sigma(R_c, c'), c'') = \sigma(R_c, c'')$ for $c \geq c' \geq c''$; as one would expect from the intuitive notion of filtering.

The formulation of filtering given here is simpler than the definition given in [11, 13, 15] (and subsequently adopted by SeaView [17]). The main difference is that the null-subsumption property of [11, 13, 15] is no longer being required (principally because the null-integrity property of [11, 13, 15] has been dropped). In the formulation given here null values require no special treatment from a security viewpoint.

An important consequence of the inter-instance integrity property is that it allows instances such as shown in figure 3. Note that there is a C tuple whose key class is U, but the key value (and class) do not occur in any U tuple. U subjects will see an empty relation in this case, as indicated in figure 4. We will see in section 5 that this phenomenon has significant, and beneficial, implications for referential integrity. Contrast figure 3 with the instance shown in figure 5 (with the Unclassified view shown in figure 6). With our definition of inter-instance integrity both figures 3 and 5 are valid Confidential instances of SOD, but they are semantically different.[4] We will return to consideration of this issue in section 5.

Next, we have the following polyinstantiation integrity constraint which prohibits polyinstantiation within a single access class.

Property 3 [Polyinstantiation Integrity (PI)] A multilevel relation R is said to satisfy polyinstantiation integrity (PI) if and only if for every R_c we have for all A_i that $A_1, C_1, C_i \rightarrow A_i$. □

[4] Note that with prior definitions of inter-instance integrity [11], which include null-subsumption, the closest one can get to these instances is to have the C instance of figure 3 with corresponding U instance of figure 6.

SHIP		OBJ		DEST		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Mining	C	Sirius	C	C
Enterprise	U	Spying	S	Rigel	S	S
Enterprise	U	Coup	TS	Orion	TS	TS

Figure 1: A multilevel relation SOD

SHIP		OBJ		DEST		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Mining	C	Sirius	C	C

Figure 2: Confidential view of figure 1

SHIP		OBJ		DEST		TC
Enterprise	U	Mining	C	Sirius	C	C

Figure 3: Another Confidential Instance of SOD

SHIP	OBJ	DEST	TC

Figure 4: Unclassified view of figure 3

SHIP		OBJ		DEST		TC
Enterprise	U	null	U	null	U	U
Enterprise	U	Mining	C	Sirius	C	C

Figure 5: Yet Another Confidential Instance of SOD

SHIP		OBJ		DEST		TC
Enterprise	U	null	U	null	U	U

Figure 6: Unclassified view of figure 5

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Spying	U	Rigel	S	S

Figure 7: Violation of Polyinstantiation Integrity

This property stipulates that the user-specified apparent key A_1, in conjunction with the classification attributes C_1 and C_i, functionally determines the value of the attribute A_i. In other words the real primary key of the relation is $A_1, C_1, C_2, \ldots, C_n$. This formulation of PI was first proposed in [11].[5] The effect of polyinstantiation integrity is to rule out instances such in figure 7, where there are two values labeled U for the Objective attribute of the Enterprise.

Finally, we introduce the fourth integrity property, which was first identified in [19]. The intuitive idea is that every entity in a relation can have at most one tuple for every access class.[6] The requirement is formally as follows.

Property 4 [PI-tuple-class] R satisfies tuple-class polyinstantiation integrity if and only if for every instance R_c, $(\forall A_i \notin A_1)[A_1, C_1, TC \rightarrow A_i]$. □

To appreciate the motivation for PI-tuple-class consider the instance $\mathrm{SOD_U}$ given in figure 8. Let Starship be the apparent key of this relation. Eight instances of $\mathrm{SOD_S}$ are shown in figure 9. All these instances of $\mathrm{SOD_S}$ are consistent with $\mathrm{SOD_U}$ of figure 8 with respect to the inter-instance integrity property. In other words, if tuples with TC = S are removed from any of the eight $\mathrm{SOD_S}$ instances we are left with the single tuple of the $\mathrm{SOD_U}$ instance. Moreover, all eight instances of $\mathrm{SOD_S}$ satisfy the entity integrity and polyinstantiation integrity properties. Thus any of these eight instances are acceptable under properties 1, 2 and 3.

It is clear that instances 2, 3 and 4 of figure 9 have a much simpler interpretation than instances 5, 6, 7 and 8. The PI-tuple-class property formalizes this intuitive distinction by requiring that there be at most one tuple for the Enterprise at each access class. Instances 2, 3 and 4 have exactly one S tuple for the $<$Enterprise, U$>$, in addition to the single U tuple. The U tuple is then easily interpreted to denote a cover story with respect to the S tuple. Instances 5, 6, 7 and 8 are in violation of PI-tuple-class because they all have two or more tuples with tuple class S which have the same apparent key and key class (i.e., $<$Enterprise, U$>$).

Polyinstantiation integrity (or PI) and PI-tuple-class are independent properties. Instances 5, 6, 7 and 8 of figure 9 illustrate relation instances which satisfy PI but not PI-tuple-class. The instance of $\mathrm{SOD_S}$ given in figure 10 shows how PI-tuple-class can be satisfied while PI is violated.

We regard properties 3 and 4 as the formal definition of the informal notion of A_1 as the user-specified apparent primary key. Note that for single level relations C_1 and C_i will be equal to the same constant value in all tuples. In this case property 3 amounts to saying $A_1 \rightarrow A_i$, which is precisely the definition of primary key in standard relational theory. Similarly, property 4 also reduces to $A_1 \rightarrow A_i$ for single-level relations.

3 SEMANTICS OF POLYINSTANTIATION

In the previous section we have given a formal model (albeit without referential integrity) for multilevel relations with element-level labeling. In this section we consider the semantic interpretation of polyinstantiation in these relations. The essential points can be illustrated in context of the instance of figure 11. This instance is permitted by the integrity properties of section 2. It exhibits two distinct forms of polyinstantiation which we call *entity polyinstantiation* and *element polyinstantiation*.

Entity polyinstantiation arises when there are two tuples with the same value of the apparent primary key, but with different values of the key class. This is illustrated in figure 11 where the

[5] It should be noted that the SeaView definition of polyinstantiation integrity [16, 17] requires property 3, but in addition requires a multi-valued dependency property which has the undesirable consequence of introducing spurious tuples in the multilevel relation [11].

[6] The formulation of this property in [19] disclosed some problems with this intuitive idea, which have been carefully avoided in the present paper. We also note that the behavior of multilevel relations in LDV [10] essentially requires this property, although the precise formalization and detailed semantics are somewhat different.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

Figure 8: An instance SOD_U

No.	Starship		Objective		Destination		TC
1	Enterprise	U	Exploration	U	Talos	U	U
2	Enterprise	U	Exploration	U	Talos	U	U
	Enterprise	U	Spying	S	Talos	U	S
3	Enterprise	U	Exploration	U	Talos	U	U
	Enterprise	U	Exploration	U	Rigel	S	S
4	Enterprise	U	Exploration	U	Talos	U	U
	Enterprise	U	Spying	S	Rigel	S	S
5	Enterprise	U	Exploration	U	Talos	U	U
	Enterprise	U	Exploration	U	Rigel	S	S
	Enterprise	U	Spying	S	Rigel	S	S
6	Enterprise	U	Exploration	U	Talos	U	U
	Enterprise	U	Spying	S	Talos	U	S
	Enterprise	U	Spying	S	Rigel	S	S
7	Enterprise	U	Exploration	U	Talos	U	U
	Enterprise	U	Spying	S	Talos	U	S
	Enterprise	U	Exploration	U	Rigel	S	S
8	Enterprise	U	Exploration	U	Talos	U	U
	Enterprise	U	Spying	S	Talos	U	S
	Enterprise	U	Exploration	U	Rigel	S	S
	Enterprise	U	Spying	S	Rigel	S	S

Figure 9: Eight instances of SOD_S

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Spying	U	Rigel	S	S

Figure 10: An instance of SOD_S satisfying PI-tuple-class but not PI

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Spying	S	Rigel	S	S
Enterprise	S	Attack	S	Sirius	S	S

Figure 11: Entity and Element Polyinstantiation

Starship		Objective		Destination		TC
Enterprise	U	null	U	null	U	U
Enterprise	U	Spying	S	Rigel	S	S

Figure 12: An S instance of SOD

Starship		Objective		Destination		TC
Enterprise	U	null	U	null	U	U

Figure 13: The U view of figure 12

Starship		Objective		Destination		TC
Enterprise	U	Spying	S	Rigel	S	S

Figure 14: Another S instance of SOD

Starship	Objective	Destination	TC

Figure 15: The U view of figure 14

third tuple has the same apparent key value (i.e., Enterprise) as the first (or second) tuple, but the key class in the third tuple (i.e., S) is different from the key class in the first (or second) tuple (i.e., U). The interpretation is that in this case there are two Starships, the $<$ Enterprise, U $>$ and the $<$ Enterprise, S $>$. In other words the two S-tuples pertain to two distinct real world entities. In contrast, the top two tuples in figure 11 refer to the same starship $<$ Enterprise, U $>$; the S-tuple gives the classified values for the Objective and Destination attributes, whereas the U-tuple gives the unclassified cover story for both attributes. The S-tuple for $<$ Enterprise, S $>$ pertains to a completely different Starship whose existence is not known at the unclassified level. In short, entity polyinstantiation is interpreted by asserting that a real-world entity is identified in the database by the apparent key and key class.

Element polyinstantiation, on the other hand, arises when there are two tuples with the same value of the apparent primary key, and with the same value of the key class. This is illustrated in figure 11 by the first two tuples. The interpretation, in this case, is that both tuples refer to the same Starship in the real world, viz., the $<$Enterprise, U$>$. The U-tuple gives the unclassified values for the Objective and Destination attributes, whereas the S-tuple gives the classified values for these attributes. In short, element polyinstantiation is interpreted by asserting that the same real-world entity has different values for its attributes at different access classes.

Figures 12 through 15 further illustrate a subtle aspect of the inter-instance property, briefly alluded to in the previous section. Figure 12 shows element polyinstantiation for a single Starship called Enterprise, whose key class is U. Even though the values of the Objective and Destination attributes in the U tuple are null, we will consider this to be element polyinstantiation because non-null values have been given in the S tuple. The corresponding U instance is shown in figure 13. Now consider the S instance of SOD shown in figure 14. This instance is allowed by the integrity properties of the previous section. The corresponding U instance is shown in figure 15. Note that even though the S tuple of figure 14 has a component labeled U, the U instance is completely empty.

What interpretation are we to give to the fact that the Starship name is labeled U in figure 14? We will understand such a situation to mean that the Enterprise may become visible at the U level, even though currently it is not. The implication is that if a U tuple for the <Enterprise, U> does come about in SOD, it is going to refer to exactly the same real-world starship that the existing S tuple refers to.

We will see, in section 5, that this interpretation turns out—rather unexpectedly—to be important for certain aspects of referential integrity. It should be kept in mind that, if the semantics of the application dictate that the instance of figure 14 is not allowed we can prevent its occurrence by the usual integrity constraints in relational systems. The point is that our data model does not inherently rule out this instance, as is done by previous data models [6, 11, 12, 13, 15, 16, 17, 19, 20] in this area.

4 PRIOR WORK ON REFERENTIAL INTEGRITY

In this section we review previous work on referential integrity and point out its weaknesses. The notion of a foreign key relates two relations: a *referencing* relation, say R, and a *referenced* relation, say Q. A foreign key FK of R is declared to be one or more attributes of R which collectively reference the primary key PK of Q. The number of attributes in FK and PK, as well as their domains (such as number or character string), must be identical for a valid declaration of a foreign key.

The first requirement for foreign keys is as follows.

Property 5 [Foreign Key Integrity] Let FK be a foreign key of the referencing relation R. A multilevel relation R satisfies foreign key integrity if and only if for all instances R_c and $t \in R_c$

1. Either $(\forall A_i \in FK)[t[A_i] = \text{null}]$ or $(\forall A_i \in FK)[t[A_i] \neq \text{null}]$.
2. $A_i, A_j \in FK \Rightarrow t[C_i] = t[C_j]$ (i.e., FK is uniformly classified). □

The first part of this property arises from standard relations. The motivations for the second part of this property are similar to those for the uniform classification of apparent primary keys in the entity integrity property.

The foreign key property by itself is not sufficient. In standard relations, the referential integrity property precludes the possibility of dangling references from R to Q. In other words a non-null foreign key must have a matching tuple in the referenced relation. To avoid signaling channels that arise due to upward (or sideways) references, SeaView originally proposed the following formulation of referential integrity for multilevel relations [6].

Property 6 [Referential Integrity (SeaView I)] Let FK be a foreign key of the referencing relation R. Let Q be the referenced relation, with apparent primary key AK. R and Q satisfy referential integrity if and only if for all instances R_c and Q_c occurring together, and for all $t \in R_c$ such that $t[FK] \neq \text{null}$, there exists $q \in Q_c$ such that $t[FK] = q[FK] \wedge t[C_{FK}] \geq q[C_{AK}]$. □

Unfortunately, the above formulation results in referential ambiguity. The problem of referential ambiguity was first noted by Gajnak [9]. It is illustrated in figures 16(a), where SOD is as before, and CAPTAIN is the apparent primary key of the CS relation. In this example SHIP is a foreign key from CS to SOD. In the CS relation, at the U level Kirk has not been assigned to any starship, while at the S level Kirk's assignment is to the Enterprise. However, due to entity polyinstantiation, there are two starships called Enterprise in SOD. It is therefore ambiguous as to which one Kirk is assigned to (or perhaps he is captain of both).

Gajnak's observations led SeaView researchers to modify the above referential integrity property to require equality of the key classifications [16, 17], as follows.

SHIP		OBJ		DEST		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	S	Spying	S	Rigel	S	S

CAPTAIN		SHIP		TC
Kirk	U	null	U	U
Kirk	U	Enterprise	S	S

(a)

SHIP		OBJ		DEST		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Spying	S	Rigel	S	S

CAPTAIN		SHIP		TC
Kirk	U	null	U	U
Kirk	U	Enterprise	S	S

(b)

Figure 16: Foreign key references from CS to SOD

Property 7 [Referential Integrity (SeaView II)] Let FK be a foreign key of the referencing relation R. Let Q be the referenced relation, with apparent primary key AK. R and Q satisfy referential integrity if and only if for all instances R_c and Q_c occurring together, and for all $t \in R_c$ such that $t[FK] \neq \text{null}$, there exists $q \in Q_c$ such that $t[FK] = q[FK] \wedge t[C_{FK}] = q[C_{AK}]$. □

This formulation takes care of referential ambiguity, but has the unfortunate consequence of curtailing the modeling power of multilevel relations. For example, the instance of figure 16(b) is not valid anymore. However, there is nothing semantically incorrect with these relations. We are simply trying to keep the assignment of Kirk to the Enterprise secret, whereas the existence of the Enterprise is unclassified. If we store information about starships and about assignment of captains in two different relations, the SeaView II rule will allow us to keep the assignment of Kirk secret only if it is to a secret starship. We cannot classify the assignment of Kirk to an unclassified starship!

5 PROPOSED SEMANTICS OF REFERENTIAL INTEGRITY

Prior work on referential integrity in multilevel relations leaves us in an impasse. We either have referential ambiguity or substantial loss of modeling power. Since neither of these is a viable alternative, we must find some means of getting around this impasse.

The problem of referential ambiguity arises due to entity polyinstantiation. Therefore our proposal is to retain the original SeaView referential integrity property (i.e., property 6) which allows downward references,[7] and disallow entity polyinstantiation. Let us see how entity polyinstantiation can be securely prevented.[8] We distinguish two kinds of relations for this purpose, as follows.

[7]We will see later in this section that property 6 needs to be slightly modified to work correctly.

[8]Note that element polyinstantiation can also be securely prevented using the technique of [20]. Our proposal is to

- *Atomic Relations*: In these relations the apparent primary key AK does not contain a foreign key as a proper subset of the attributes of AK.
- *Composite Relations*: In these relations the apparent primary key AK does contain a foreign key as a proper subset of the attributes of AK.

These two cases are respectively discussed in the following two subsections.

5.1 Prevention of Entity Polyinstantiation in Atomic Relations

The basic technique for preventing entity polyinstantiation in atomic relations is to partition the domain of the primary key among the various security classes possible for the primary key [14].[9] For our SOD example, we can introduce a new attribute, called Starship#. Whenever a new tuple is inserted, we enforce the requirement that all unclassified Starships are numbered between 1 and 1,000, all confidential Starships are numbered between 1,001 and 2,000, and so on.

In a SQL-like data definition language, the modified SOD schema could be created as follows:

```
CREATE TABLE SOD
  ( Starship#    SMALL INTEGER NOT NULL [U:TS]
    Starship     CHAR(15) NOT NULL [U:TS]
    Objective    CHAR(15) {U, TS},
    Destination  CHAR(20) [U:TS],
    Primary Key (Starship# ),
    CHECK (Subject Access class = 'U'  AND Starship# BETWEEN 1    AND 1000),
    CHECK (Subject Access class = 'C'  AND Starship# BETWEEN 1001 AND 2000),
    CHECK (Subject Access class = 'S'  AND Starship# BETWEEN 2001 AND 3000),
    CHECK (Subject Access class = 'TS' AND Starship# BETWEEN 3001 AND 4000) );
```

The notation [L:H] specifies a range of security classes with lower bound L and upper bound H. The notation {X,Y,Z} enumerates the allowed values for the security class as one of X, Y or Z. Here, the domain of the security class of the apparent primary key Starship# has been specified as a range with a lower bound of U and an upper bound of TS. However, the domain of the Starship# has been partitioned across these security classes.

It should be noted that confidentiality does not depend on correct partitioning of the key space by the integrity enforcement mechanism of the Database Management System (DBMS). If this mechanism fails, or is deliberately malicious due to Trojan Horse infection, the integrity properties will fail but there will be no leakage of information. To fully substantiate this statement, we would need to give a kernelized implementation of the DBMS, i.e., an implementation which does not use subjects exempted from the mandatory controls of the underlying multilevel secure operating system. Description of such an implementation is outside the scope of this paper.

5.2 Prevention of Entity Polyinstantiation in Composite Relations

Consider the relations shown in figure 17. SOD is the familiar relation, with apparent primary key SHIP. Let CAPTAIN be the apparent primary key of the relation CR. Now consider the relation CSH, some of whose instances are illustrated in figure 18. The apparent primary key of CSH consists of the attributes CAPTAIN and SHIP. By the entity integrity property (property 1) both attributes must be uniformly classified. Hence only one classification is shown for these two attributes. Suppose

eliminate entity polyinstantiation as part of the data model, but keep element polyinstantiation as a possible option.

[9] This is analogous to the manner in which static resource allocation across security classes eliminates covert channels which arise due to dynamic resource allocation in multilevel Operating Systems.

CAPTAIN is a foreign key from CSH to CR, and SHIP is a foreign key from CSH to SOD. For the rest of this discussion, assume that SOD and CR are as shown in figure 17.

A valid instance of CSH is shown in figure 18(a). The top two tuples in figure 18(a) correspond to the same entity, viz., $<$Kirk, Enterprise, U$>$, and indicate the occurrence of element polyinstantiation. The interpretation is that Kirk is assigned to the Enterprise for 15 hours at the U level, and for 10 hours at the S level. The bottom three tuples of figure 18(a) correspond to three distinct entities, all of which are secret. These three entities represent the assignment of Kirk to Voyager, and the assignments of Spock to the Enterprise and to the Voyager. These entities are labeled S because each one of them references a secret entity in SOD or CR or both. Since only downward references are allowed, by property 7 these foreign keys must be labeled S.

Figures 18(b) and (c) illustrate the phenomenon of entities which are not currently visible at a lower level, but may become visible in the future. This situation was encountered in context of the inter-instance property in section 2, and was also discussed in the latter part of section 3. We now see that this phenomenon is useful in relations which relate existing entities. The single S tuple in figure 18(b) assigns Kirk to the Enterprise with a secret load of 10 hours/week. It is possible that later Kirk is assigned to the Enterprise with a unclassified load of 15 hours/week, as shown in figure 18(c). Note that in going from figure 18(b) to (c), from a S subject's point of view, we are not instantiating another entity but merely making an unclassified entity visible at the unclassified level. From a U subject's point of view, we are instantiating another entity at the U level, but this entity may or may not have previously instantiated at a higher level.

Figures 18(d) and (e) illustrate the incorrect approach to handling the situation of figures 18(b) and (c). In this case the S tuple in figure 18(d) is for the entity $<$Kirk, Enterprise, S$>$. This opens up the possibility of entity polyinstantiation as shown in figure 18(e). References from some other relation to $<$ Kirk, Enterprise $>$ in CSH will therefore be ambiguous. In such cases we must make sure that we do not over classify the apparent primary key of CSH.

5.3 Referential Integrity Property

Based on our discussion we recommend going back to the original formulation of the SeaView referential integrity property (i.e., property 6). We need to change this property slightly to avoid references to entities that are potentially visible at level c, but are currently only instantiated at levels above c. This requires the additional condition, $t[C_{FK}] \geq q[TC]$, relative to property 6, giving us the following definition.

Property 8 [Referential Integrity] Let FK be a foreign key of the referencing relation R. Let Q be the referenced relation, with apparent primary key AK. R and Q satisfy referential integrity if and only if for all instances R_c and Q_c occurring together, and for all $t \in R_c$ such that $t[FK] \neq$ null, there exists $q \in Q_c$ such that $t[FK] = q[FK] \wedge t[C_{FK}] \geq q[C_{AK}] \wedge t[C_{FK}] \geq q[TC]$. □

With this definition, and with elimination of entity polyinstantiation, we will have eliminated referential ambiguity while retaining the expressive power to allow classification of relationships among unclassified entities. Elimination of entity polyinstantiation can be formally expressed as follows.

Property 9 [No Entity Polyinstantiation] A multilevel relation R is said to satisfy the "no entity polyinstantiation" property if and only if for every R_c we have $A_1 \rightarrow C_1$. □

6 CONCLUSION

In this paper we have shown that previous work on referential integrity leaves us with a choice of either accepting referential ambiguity or severely curtailing the modeling power of multilevel

SHIP		OBJ		DEST		TC
Enterprise	U	Exploration	U	Talos	U	U
Voyager	S	Spying	S	Rigel	S	S

CAPTAIN		RANK		TC
Kirk	U	Admiral	U	U
Spock	S	General	S	S

Figure 17: Relations SOD and CR

CAPTAIN	SHIP		HOURS/WEEK		TC
Kirk	Enterprise	U	15	U	U
Kirk	Enterprise	U	10	S	S
Kirk	Voyager	S	30	S	S
Spock	Enterprise	S	20	S	S
Spock	Voyager	S	15	S	S

(a)

CAPTAIN	SHIP		HOURS/WEEK		TC
Kirk	Enterprise	U	10	S	S

(b)

CAPTAIN	SHIP		HOURS/WEEK		TC
Kirk	Enterprise	U	15	U	U
Kirk	Enterprise	U	10	S	S

(c)

CAPTAIN	SHIP		HOURS/WEEK		TC
Kirk	Enterprise	S	10	S	S

(d)

CAPTAIN	SHIP		HOURS/WEEK		TC
Kirk	Enterprise	U	15	U	U
Kirk	Enterprise	S	10	S	S

(e)

Figure 18: Foreign key references from CSH to SOD and CR

relations. We have shown how to escape this impasse by eliminating entity polyinstantiation, while retaining element polyinstantiation (as an option).

In future work, one should define a formal update semantics for relations which satisfy the core integrity properties of section 2, and the referential integrity and "no entity polyinstantiation" properties of section 5. Completeness and soundness of the semantics should be proved. It is also important to develop correct decomposition and recovery algorithms for a kernelized architecture (i.e., an architecture in which no subject is exempted from the simple-security or star-properties) which give these semantics.

References

[1] "Multilevel Data Management Security," Committee on Multilevel Data Management Security, Air Force Studies Board, National Research Council, Washington, DC (1983).

[2] Bell, D.E. and LaPadula, L.J. "Secure Computer Systems: Unified Exposition and Multics Interpretation." MTR-2997, MITRE (1975).

[3] Burns, R.K. "Referential Secrecy." *IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, 133-142.

[4] Date, C.J. *An Introduction to Database Systems.* Volume II, Addison-Wesley, (1983).

[5] Denning, D.E., Lunt, T.F., Schell, R.R., Heckman, M., and Shockley, W.R. "A Multilevel Relational Data Model." *Proc. IEEE Symposium on Security and Privacy*, 220-234 (1987).

[6] Denning, D.E., Lunt, T.F., Schell, R.R., Shockley, W.R. and Heckman, M. "The SeaView Security Model." *Proc. IEEE Symposium on Security and Privacy*, 218-233 (1988).

[7] Doshi, V.M. and Jajodia, S. "Referential Integrity in Multilevel Secure Database Management Systems." *Proceedings of the IFIP TC 11 8th International Conference on Information Security*, (1992).

[8] Doshi, V.M. and Jajodia, S. "Enforcing Entity and Referential Integrity in Multilevel Databases." *Proc. 15th NIST-NCSC National Computer Security Conference*, Baltimore, MD, October 1992, pages 134-143.

[9] Gajnak, G.E. "Some Results from the Entity-Relationship Multilevel Secure DBMS Project." *Aerospace Computer Security Applications Conference*, 66-71 (1988).

[10] J. Thomas Haigh, Richard C. O'Brien, and Daniel J. Thomsen, "The LDV Secure Relational DBMS Model." *Database Security IV: Status and Prospects*, S. Jajodia and C. E. Landwehr (editors), North-Holland, 1991, pages 265-279.

[11] Jajodia, S. and Sandhu, R.S. "Polyinstantiation Integrity in Multilevel Relations." *Proc. IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, pages 104-115.

[12] Jajodia, S. and Sandhu, R.S. "Polyinstantiation Integrity in Multilevel Relations Revisited." *Database Security IV: Status and Prospects*, Jajodia, S. and Landwehr, C. (editors), North-Holland, pages 297-307, 1991.

[13] Jajodia, S. , Sandhu, R.S., and Sibley E. "Update Semantics of Multilevel Relations." *Proc. 6th Annual Computer Security Applications Conf.*, Tucson, AZ, December 1990, pages 103-112.

[14] Jajodia, S. and Sandhu, R.S. "Enforcing Primary Key Requirements in Multilevel Relations," *Proc. 4th RADC Workshop on Multilevel Database Security*, Rhode Island, April 1991.

[15] Jajodia, S. and Sandhu, R.S. "A Novel Decomposition of Multilevel Relations Into Single-Level Relations." *Proc. IEEE Symposium on Security and Privacy*, Oakland, California, May 1991.

[16] Lunt, T.F. et al. *Secure Distributed Data Views.* Volume 1-4, SRI Project 1143, SRI International (1988-89).

[17] Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M. and Shockley, W.R. "The SeaView Security Model." *IEEE Transactions on Software Engineering*, 16(6):593-607 (1990).

[18] Lunt, T.F. and Hsieh, D. "Update Semantics for a Multilevel Relational Database." *Database Security IV: Status and Prospects*, Jajodia, S. and Landwehr, C. (editors), North-Holland, pages 281-296, 1991.

[19] Sandhu, R.S., Jajodia, S. and Lunt, T. "A New Polyinstantiation Integrity Constraint for Multilevel Relations." *Proc. IEEE Workshop on Computer Security Foundations*, Franconia, New Hampshire, June 1990, pages 159-165.

[20] Sandhu, R.S. and Jajodia, S. "Honest Databases That Can Keep Secrets." *14th NIST-NCSC National Computer Security Conference*, Washington, D.C., October 1991, pages 267-282.

[21] Sandhu, R.S. and Jajodia, S. "Polyinstantiation for Cover Stories." *Proc. European Symposium on Research in Computer Security*, Toulouse, France, November 1992, pages 307-328. Published as *Lecture Notes in Computer Science, Vol 648, Computer Security—ESORICS92* (Deswarte, Y., Eizenberg, G., and Quisquater, J.-J., editors), Springer-Verlag, 1992.