Proceedings of IEEE Computer Security Foundations Workshop VI, Franconia, NH, June 1990, pages 159-165.

A New Polyinstantiation Integrity Constraint For Multilevel Relations

Ravi Sandhu * , Sushil Jajodia*

Department of Information Systems
and Systems Engineering
George Mason University
Fairfax, VA, 22030-4444

Teresa Lunt†

Computer Science laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA, 94025

Abstract

We propose a new polyinstantiation integrity constraint for multilevel relations based on the intuitive idea that every entity in a relation can have at most one tuple for every access class. We discuss the consequences of this property and some of its variations.

1 INTRODUCTION

In multilevel environments it is inherent that users with different clearances see different sets of facts. This inevitably results in polyinstantiation, i.e., the simultaneous existence of data objects, attributes or values which are indistinguishable except for classification. Polyinstantiation significantly complicates the meaning of multilevel databases relative to ordinary single level databases. The best we can do is to give as simple a semantics for polyinstantiation as feasible. This paper describes the results of our efforts at formulating a particularly simple polyinstantiation property for multilevel relations.

To appreciate the fundamental importance of polyinstantiation integrity constraints consider the multilevel relation SOD shown in table 1. The schema specifies that the data elements are individually labeled in the range U to S, i.e., we have element level labeling. In addition each tuple has a composite tuple-class label, denoted by TC, which is defined to be the least upper bound of the labels on the individual data elements of that tuple. The Starship attribute is designated as the primary key of this relation. This means that all tuples with the same value *and* classification for the Starship attribute pertain to the same entity in the external world. Eight possible instances of SOD, at the S level, are shown in table 1(c). Each instance describes a single starship, viz., the Enterprise/U. Excepting the degenerate case of instance 1, these instances give different information regarding the objective and destination of the Enterprise to U and S users.

We will be using the relation SOD for all our examples in this paper. In many cases we will only use the extreme points U and S of the security lattice of table 1(b). The incomparable labels are required for some of the more subtle points.

Let us describe multiple tuples for the same real world entity as polyinstantiated tuples. Polyinstantiation integrity is concerned with controlling the legitimate manner in which polyinstantiated tuples can arise. Now the SeaView model [1, 4] can accommodate either instances 1, 2, 3, 8 or 1, 4 of table 1(c) within a single schema. In [3] it is argued that all eight of these instances have realistic and useful interpretations. Therefore a general multilevel model should accommodate all eight within a single schema. It was shown in [3] that this can be accomplished by relaxing the polyinstantiation constraints of SeaView. Let us call the model developed in [3] as the Oakland model. During subsequent discussions among the three authors of this paper it became clear that in many practical situations it is most useful to have relation schemas which allow instances 1, 2, 3, 4 while ruling out 5, 6, 7, 8.

*This research was supported (partially) by the Center for Excellence in Command, Control, Communications, and Intelligence at George Mason University. The Center's general research program is sponsored by the Virginia Center for Innovative Technology, MITRE Corporation, the Defense Communications Agency, CECOM, PRC/ATI, ASD (C3I), TRW, AFCEA, and AFCEA NOVA.

†This research was supported (partially) by the U. S. Air Force, Rome Air Development Center (RADC), who funded SRI through subcontract R009406 with IITRI (U. S. Government contract F30602-87-D-0094).

This naturally led to the question: what polyinstantiation constraints will admit instances 1, 2, 3, 4 while ruling out 5, 6, 7, 8? The intuitive idea is that every entity in a relation can have at most one tuple for every access class. We call this the Franconia model. In this paper we formalize this constraint and discuss some of its consequences and variations.

The rest of the paper is organized as follows. Section 2 reviews the basic definitions of multilevel relations to establish the background for this paper and properties common to all three models. Our review is necessarily brief. For detailed discussion on the motivation underlying these properties see [1, 2, 3, 4]. Sections 3 and 4 review the polyinstantiation integrity constraints of the Oakland and SeaView models. Section 5 formulates our new constraint, resulting in the Franconia model. It goes on to discuss some variations of this model. Section 6 concludes the paper.

2 BACKGROUND

In this section we review the basic definitions required for what follows. The properties defined here are required of all models discussed in the paper. These properties constitute a common core to which SeaView, Oakland and Franconia make specific additions.

We assume familiarity with the basic concepts of relational theory. For our purpose a relation is defined as a subset of a Cartesian product of sets, called domains. Relational theory distinguishes between the state-independent relation scheme and state-dependent relation instances. These correspond respectively to the Cartesian product and to the specific subset of it which constitutes the relation instance in a given state.

The definition of a *multilevel relation* R similarly consists of the following two parts.

Definition 1 [MULTILEVEL RELATION SCHEME] A state-invariant multilevel relation scheme

$$R(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC)$$

where each A_i is a *data attribute* 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 $[L_i, H_i]$ which defines a sub-lattice of access classes ranging from L_i up to H_i. The domain of TC is $[\text{lub}\{L_i\}, \text{lub}\{H_i\}]$. □

Definition 2 [RELATION INSTANCES] A collection of state-dependent *relation instances*

$$R_c(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC)$$

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 $a_i \in D_i$, $c \geq c_i$ and $tc = \text{lub}\{c_i\}$. Moreover, if a_i is not null then $c_i \in [L_i, H_i]$. We require that c_i be defined even if a_i is null, i.e., a classification attribute cannot be null. □

We often write the elements of a tuple as a_1/c_1, a_2/c_2, etc. to emphasize the scope of each classification label. Since tc is computed from the other classification attributes, it is included or omitted as convenient. We use the notation $t[A_i]$ to mean the value of the A_i attribute in tuple t, and similarly for $t[C_i]$ and $t[TC]$.

The concept of primary key is critical to the relational model. Informally, the primary key is a minimal subset of the attributes whose values uniquely identify exactly one tuple in every instance of the relation(if such a tuple exists). In other words there cannot be more than one tuple with the same primary key. In relational theory primary keys are defined in terms of the more basic notion of functional dependencies. In a multilevel setting the concept of functional dependencies is itself clouded because a relation instance is now a collection of sets of tuples rather than a single set of tuples.

Rather than trying to resolve this complex issue here, we follow the lead of SeaView and assume there is a user specified primary key AK consisting of a subset of the data attributes A_i. This is called the *apparent primary key* of the multilevel relation scheme. In general AK will consist of multiple attributes. Entity integrity from the standard relational model prohibits null values for any of the attributes in AK. This property extends to multilevel relations as follows.

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

1. $A_i \in AK \Rightarrow t[A_i] \neq \text{null}$.
2. $A_i, A_j \in AK \Rightarrow t[C_i] = t[C_j]$, i.e., AK is *uniformly classified.* Define C_{AK} to be the classification of the apparent key, i.e., $t[C_i] = t[C_{AK}]$ for all $A_i \in AK$.
3. $A_i \notin AK \Rightarrow t[C_i] \geq t[C_{AK}]$. □

These requirements are quite reasonable and some intuitive justification for them is given in [1, 2]. The standard relational model also has a referential integrity property to ensure consistency of references from one relation to another. In this paper our focus is on single relations, so the multilevel analog of referential integrity is not relevant.

This brings us to our first integrity property which is specific to multilevel relations, as opposed to being an analog of some similar property for single level relations. It is concerned with the consistency between relation instances at different access classes. The requirement is expressed in terms of the following function.

Definition 3 [Filter Function] Given the c-instance R_c of a multilevel relation the filter function σ produces the c'-instance $R_{c'} = \sigma(R_c, c')$ for $c' \leq c$ as follows: for every tuple $t \in R_c$ such that $t[C_{AK}] \leq c'$ there is a tuple $t' \in R_{c'}$ with

$$\begin{aligned} t'[AK, C_{AK}] &= t[AK, C_{AK}] \\ \text{and for } i \notin AK & \\ t'[A_i, C_i] &= \begin{cases} t[A_i, C_i] & \text{if } t[C_i] \leq c' \\ < \text{null}, t[C_{AK}] > & \text{otherwise} \end{cases} \end{aligned}$$

There are no tuples in $R_{c'}$ other than those derived by the above rule. □

It is evident that $\sigma(R_c, c) = R_c$ and for $c'' < c' < c$, $\sigma(\sigma(R_c, c'), c'') = \sigma(R_c, c'')$, as one would expect from the intuitive notion of filtering. Consistency of relation instances at different access classes is now easily stated as follows.

Property 2 [Inter-Instance Integrity] R satisfies inter-instance integrity if and only if for all states and all $c' \leq c$ we have $R_{c'} = \sigma(R_c, c')$. □

At this point it is important to clarify the semantics of null values, particularly regarding the subsumption of null values by non-null ones. To this end we have the following definition.

Definition 4 [Subsumption] Tuple t *subsumes* tuple s if for every attribute A_i, either $t[A_i, C_i] = s[A_i, C_i]$ or $s[A_i] = \text{null}$ and $t[A_i] \neq \text{null}$.* □

That is, t subsumes s if they agree everywhere except possibly for some attributes where s is null and t non-null, independent of classification.

*It is possible to define a stricter notion of subsume by additionally requiring $s[C_i] = t[C_i]$ in the latter case where $s[A_i] = \text{null}$. This strict notion of subsume is identical to that in single level relations, i.e., data and classification attributes are not distinguished. The issue of which subsumption definition to adopt is more subtle and controversial than may appear at first sight. For the sake of uniformity we have chosen to use definition 4 throughout this paper.

Property 3 [Subsumption Integrity] All multilevel relation instances are made *subsumption free* by exhaustive elimination of subsumed tuples. □

Subsumption of null values is required for example to have σ produce the following U-instance from S-instances 2 through 8 of table 1(c).

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

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

Property 4 [PI-FD] R satisfies FD polyinstantiation integrity if and only if for every R_c we have for all A_i

$$AK, C_{AK}, C_i \rightarrow A_i$$ □

This property stipulates that the user-specified apparent key AK, in conjunction with the classification attributes C_AK and C_i, functionally determines the value of the A_i attribute.

We regard property 4 as the formal definition of the informal notion of AK as the user specified primary key. Note that for single level relations C_{AK} and C_i will be equal to the same constant value in all tuples. In this case PI-FD amounts to saying $AK \rightarrow A_i$, which is precisely the definition of primary key in relational theory. The effect of PI-FD is to rule out instances such as the following, where there are two values labeled U for the Objective attribute of the Enterprise/U.

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

We reiterate that the three models discussed in this paper require properties 1 through 4, i.e., entity integrity, inter-instance integrity, subsumption integrity and PI-FD. We call these the *core properties.* Each model imposes an additional PI-constraint as described in the following sections.

3 THE OAKLAND MODEL

The Oakland model [3] admits all eight S-instances of SOD enumerated in table 1. A realistic interpretation for each of these instances, particularly 5, 6, 7 and 8, has been given in [3] at considerable length. Oakland has the following PI constraint in addition to PI-FD.

Property 5 [PI-null] R satisfies null polyinstantiation integrity if and only if for every instance R_c, we have for all $t, t' \in R_c$ such that $t[AK, C_{AK}] = t'[AK, C_{AK}]$:

$$(\forall A_i \notin AK)[t[A_i] = \text{null} \Leftrightarrow t'[A_i] = \text{null}] \qquad \square$$

In words, two polyinstantiated tuples for the same entity must either both be null or both non-null in any given non-key attribute, independent of the access class. Note that for single level relations PI-null is trivially true due to the PI-FD requirement that $AK \rightarrow A_i$.

To appreciate the significance of PI-null consider table 2. Let instance 1 and 2 respectively be the M_1 and M_2 instances of SOD in some state. Assume that no additional data is revealed at the S level. One might therefore expect the lower level information at M_1 and M_2 to uniquely determine the higher level instance at S. However, the core properties allow us either instance 3 or 4 as acceptable S instances consistent with the lower level instances 1 and 2. PI-null resolves this ambiguity by ruling out instance 4 as invalid. A similar phenomenon also shows up in table 2. Let instance 1 be the U instance of SOD. The core properties allow us either instance 2 or 3 as acceptable S instances consistent with instance 1. The ambiguity is less compelling in this situation because we do have new data, viz., Rigel/S, revealed at the S level. At any rate, PI-null rules out instance 3 as invalid.

4 THE SEAVIEW MODEL

The SeaView model [1, 4] requires the following PI constraint in addition to PI-FD.

Property 6 [PI-MVD] R satisfies MVD polyinstantiation integrity if and only if for every instance R_c, for all $A_i \notin AK$ we have

$$AK, C_{AK} \rightarrow\rightarrow A_i, C_i \qquad \square$$

The double arrow signifies multi-valued dependency. PI-MVD amounts to saying that for a given entity (i.e., given AK, C_{AK}) we should have one tuple for every combination of labeled values for the remaining attributes. Note that for single level relations PI-MVD reduces to $AK \rightarrow\rightarrow A_i$. This is vacuously true due to the PI-FD requirement that $AK \rightarrow A_i$.

In the context of table 1, SeaView only admits instances 1, 2, 3 and 8. SeaView can allow another combination of instances, viz., 1 and 4, if the Objective and Destination attributes are declared in the schema to be uniformly classified. The significant point is that SeaView cannot accommodate instances 1, 2, 3 and 4 within a single relation schema.

In terms of tables 2 and 3 SeaView makes the same choices as Oakland, i.e., respectively ruling out instances 4 and 3. In fact we have the following result showing that PI-MVD is a stronger restriction than PI-null.

Theorem 1 PI-MVD $\Rightarrow$ PI-null, but not vice versa.

Proof: Consider a relation R which satisfies PI-MVD. Let t and t' be two tuples in R which have the same values for AK, C_{AK} with $t[A_i, C_i] = \text{null}/C_{AK}$ and $t'[A_i, C_i] = a_i/c_i$ where a_i is non-null. This can happen only if $A_i \notin AK$, i.e., there is at least one non-key attribute in the relation. If A_i is the only non-key attribute then t' subsumes t. If the relation has more than one non-key attribute then by PI-MVD there must be a t'' in R such that $t''[AK, C_{AK}] = t[AK, C_{AK}]$, $t''[A_j, C_j] = t[A_j, C_j]$ for $j \neq i$ and $t''[A_i, C_i] = a_i/c_i$. But then t'' subsumes t. Therefore such a pair of tuples t and t' cannot exist in R, i.e., R satisfies PI-null. The converse is clearly not true, since PI-null allows instance 5 of table 1 while PI-MVD does not. $\square$

5 THE FRANCONIA MODEL

In this section we formulate the PI constraint which results in our new Franconia model. In the context of table 1 we had observed that SeaView will allow instances 1 and 4 if the Objective and Destination attributes are declared in the schema to be uniformly classified. In general if all the non-key attributes are uniformly classified the semantics of polyinstantiation are very simple.

The notion of *one tuple per tuple-class* is an attempt to retain some of the simplicity of uniformly classified non-key attributes, while allowing different labels for each data element. The general idea is that although we have several polyinstantiated tuples for the same entity there should be only such tuple per tuple-class. It is obvious that this would allow exactly instances 1, 2, 3 and 4 of table 1.

We state this requirement formally as follows.

Property 7 [PI-tuple-class] R satisfies tuple-class polyinstantiation integrity if and only if for every instance R_c,

$$(\forall A_i \notin AK)[AK, C_{AK}, TC \rightarrow A_i] \qquad \square$$

This is a stronger requirement than PI-FD, i.e., the above condition implies PI-FD but not vice versa.

In the rest of this section we explore some of the consequences of PI-tuple-class. Let us first go back to table 3. Instance 2 clearly violates property 7 since there are two S tuple for the Enterprise/U. So in the Franconia model we have no choice but to resort to the alternate interpretation shown in instance 3. That is, PI-tuple-class is incompatible with PI-null.

In the context of table 2 we have a different problem. Consider the tuple shown in instance 3. This tuple has all its individual data elements classified below S, yet the tuple itself is labeled S. Assume that updates are restricted to be at the access class of a user. The tuple of instance 3 therefore cannot result due to data insertion by a S user. Instead it must be materialized by the side effect of data insertions by M_1 and M_2 users. Now consider what happens if we allow such side effects in conjunction with PI-tuple-class. If the tuple of instance 3 already exists we are faced with the prospect that then a S user cannot insert the following tuple

Enterprise	U	Spying	S	Rigel	S	S

without violating PI-tuple-class. Even worse, if the above tuple already exists we would have to prevent insertion of the tuple shown in instance 3 of table 2. As argued above this insertion takes place due to side effect. Allowing or preventing this side effect depending on different circumstances is likely to complicate the simple semantics we are seeking to achieve. One possibility is to prohibit instance 3. The tuple in instance 3 has the property that its tuple-class strictly dominates the classification of the individual data elements. This can of course happen only if we have incomparable labels. To rule out such cases we can impose the following condition.

$$t[TC] > t[C_{AK}] \Rightarrow (\exists i)[t[A_i] \neq \text{null} \wedge t[C_i] = TC]$$

Next consider table 4 and the following scenario.

- Let instance 1 be the U instance, and instance 2 the M_1 instance and instance 3 the M_2 instance of a multilevel relation.

Let us say we wish to add the information at the S level that the Destination is Sirus but have no addition to the Objective information at the S level. What should the S instance look like? It has to have the U tuple of instance 1, the M_1 tuple of instance 2, and the M_2 tuple of instance 3. Given the one tuple per tuple class constraint we have to choose one of the following three S tuples.

Enterprise	U	Exploration	U	Sirus	S	S
Enterprise	U	Spying	M_1	Sirus	S	S
Enterprise	U	Coup	M_2	Sirus	S	S

Is there any application independent reason for selecting one of these tuples in preference to the others? One might attempt to give preference to polyinstantiated tuples with higher labels. For instance the first tuple above can justifiably be regarded as the "least informative" in the sense that the value of the Objective attribute is superseded at higher levels. However, since M_1 and M_2 are incomparable we still have no basis for selecting between the second and third tuples above. Moreover, if this choice is left up to the application there may still be no basis for selection. In this case PI-tuple-class would force the user to make an arbitrary selection.

We can get around this problem by refusing to explicitly show a value for the Objective attribute and instead leave it null as shon in instance 4 of table 4. Note that this null value must be labeled S, otherwise we will be in violation of PI-FD. This does require a change in our semantics of null values. The null is to be interpreted as "no additional data at this level" rather than "no available data at this level." Note that instance 4 of table 4 violates PI-null and PI-MVD, and therefore would not be admitted by either Oakland or SeaView.

6 CONCLUSION

The objective of this work is to provide insight into the question: what integrity properties should be enforced by a secure relational database management system (DBMS)? Moreover should these properties apply to for every relation, or be available for selection on a relation by relation basis as part of the schema definition?

We have identified a core set of properties which should apply to all relations. These are entity integrity, inter-instance integrity, subsumption integrity and polyinstantiation integrity in the sense of PI-FD. Specific models impose additional polyinstantiation constraints. Oakland requires PI-null, SeaView requires PI-MVD and our new Franconia model requires PI-tuple-class. Each of these properties appears likely to arise in practise often enough to justify DBMS support for its enforcement on a relation by relation basis.

Finally we have seen that some of the more subtle issues in understanding the semantics of polyinstantiation arise in the context of updates. We are convinced that a formal study of updates is required to fully answer the question raised above.

Acknowledgement

We are indebted to John Campbell, Joe Giordano, and Howard Stainer for their support and encouragement, making this work possible.

References

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

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

[3] Jajodia, S. and Sandhu, R.S. "Polyinstantiation Integrity in Multilevel Relations." *IEEE Symposium on Security and Privacy* Oakland, CA, to appear (1990).

[4] Lunt, T.F., Denning, D.E., Schell, R.R. Heckman, M. and Shockley, W.R. "Secure Distributed Data Views. Volume 2: The SeaView Formal Security Policy Model." SRI-CSL-88-15 (1989).

[5] Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M. and Shockley, W.R. "The SeaView Security Model." *IEEE Transactions on Software Engineering*, to appear.