Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Eliminating Polyinstantiation in Multilevel Relational Databases

Jin-Chou Chang and Kuen-Fang J. Jea
Institute of Computer Science
National Chung Hsing University
Taichung, Taiwan, 40227 R.O.C.

Abstract

In a relational database with multilevel relations, polyinstantiation results from avoiding a downward signaling channel. However, it also results in the cover story, data redundancy and need of update propagation. In this paper, we propose a new polyinstantiation elimination scheme to simultaneously solve the problems of the cover story, data redundancy and need of update propagation in the multilevel relation. This proposed scheme is based on a decomposition method to decompose a multilevel relation into several subrelations according to the lowest visible security level of a data attribute and whether the data attribute can be permitted to ignore the signaling channel. The algorithm and an analysis of this scheme are also presented in this paper.

1 Introduction

The multilevel relation is a consequence of imposing mandatory access control on the standard relation. Under mandatory access control, each subject/object of the system is assigned a security level. A subject can have accesses to an object according to the security level relationship between the subject and the object. In general, a subject can have read access to an object only if the security level of the subject dominates (equal to or higher than) that of the object. Moreover, to avoid revealing information to unauthorized users and destroying correctness of invisible objects through blindly writing into invisible objects, a subject can have write (update, deletion, and insertion) access to an object only if the object is classified at the subject's security level. The subject is referred to as the active entity of the system, such as user and process. The object is referred to as the passive entity containing information, such as file, record and field.

In a standard relation in databases, the tuple (record) only consists of data attributes (A_1, A_2 ... A_n), while in a multilevel relation, the tuple, as shown in Figure 1, comprises data attributes (A_1, A_2 ... A_n), their classification attributes (C_1, C_2 ... C_n) and a tuple-class attribute.

A_1	C_1	A_2	C_2	A_3	C_3	.....	TC

Figure 1. A Multilevel Relation Schema

In Figure 1, each data attribute A_i is followed by a classification attribute C_i. Each classification attribute C_i following the data attribute A_i is used to represent the security level of the data attribute A_i. The tuple-class attribute is used to represent the security level of the tuple (or entity). For each tuple, the value of the tuple-class attribute is the least upper bound of values of all classification attributes in this tuple. Furthermore, the apparent primary key AK comprising a subset of the data attributes A_i is designated by the user as the primary key.

Polyinstantiation means that users at different security levels, for a single entity in the real world, have different views. In a multilevel relation, there are two fundamental forms of polyinstantiation: entity polyinstantiation and element polyinstantiation.

- Entity polyinstantiation (polyinstantiated tuples). It means tuples with the same primary key value have different security levels associated with the primary key in a multilevel relation. For example, considering the customer relation of Figure 2 where the attribute Customer-name is the primary key, the second tuple and the third tuple result in the entity polyinstantiation. In this paper, we use a set of security levels { Top Secret (TS), Secret (S), Confidential (C), Unclassified (U)} as the domain of classification attributes and the partial order of these security levels is TS > S > C > U. TS > S means that security level TS strictly dominates security level S while TS ≥ S means that security level TS dominates security level S.

Customer-name	C_1	Customer-street	C_2	Customer-city	C_3	TC
Hayes	S	North	S	Rye	S	S
Williams	S	Park	S	Pittsfield	S	S
Williams	C	Putnam	C	Stamford	C	C

Figure 2. The Customer Relation

- Element polyinstantiation (polyinstantiated elements). It means an element (non-primary key attribute) has different security levels in two tuples with the same primary key value and key class.

Loan-number	C_1	Customer-name	C_2	Amount	C_3	TC
002125	TS	Brooks	TS	12000	TS	TS
005673	S	Adams	S	**8800**	**TS**	TS
005673	S	Adams	S	**4500**	S	S
124857	C	Curry	C	1200	C	C

Figure 3. The Borrow Relation

For example, considering the borrow relation in Figure 3, the second and third tuple result in the element polyinstantiation on the attribute **Amount**.

For a multilevel relation which has polyinstantiation, the maximum view that a user at the C′ classification (security level) can have, is generally called C′-instances. The C′-instances consist of a set of distinct tuples whose tuple-class is dominated by C′. It does not include those tuples whose tuple-class is not dominated by C′. Moreover, let $R_{C'}$ denote C′-instances and $R_{C'}$ can be derived from R_C if the security level C dominates the security level C′.

Polyinstantiation results from avoiding a downward signaling channel. However, it also results in the cover story in the multilevel relation. In the SeaView model [2], the number of tuples manufactured is exponential in the number of polyinstantiated elements. In the Jajodia-Sandhu model [4, 10, 9, 11], although the cover story is resolved, the problems of data redundancy and need of update propagation in the multilevel relation still remain unresolved. Thus, it is the motivation of this study to propose a new polyinstantiation elimination scheme which simultaneously solves the cover story, data redundancy and need of update propagation in the multilevel relation.

The remainder of this paper is organized as follows. The next section reviews how the SeaView model and the Sandhu-Jajodia model manage polyinstantiation in the multilevel relation. Section 3 introduces previous methods of eliminating entity polyinstantiation and element polyinstantiation. Section 4 describes the problems to be resolved in this study and ideas of solving the problems. Section 5 presents our polyinstantiation elimination scheme. Finally, Section 6 concludes this paper.

2 Polyinstantiation Management

In this section, we describe how the SeaView model [2, 3] and the Sandhu-Jajodia model [10, 11] manage polyinstantiation in a multilevel relation. We also point out the problems polyinstantiation gives rise to in a multilevel relation.

2.1 SeaView Model

In this model, the requirements for regulating polyinstantiation are summarized as follows:

Multilevel Entity Integrity [2]. Let AK be the set of data attributes forming the apparent primary key of a relation R. All classification attributes C_i corresponding to data attributes $A_i \in AK$ have the same value within any given tuple of R, and this class is dominated by the value of each classification attribute C_j whose data attribute $A_j \notin AK$. No tuple in an instance of R can have null values for any of the apparent primary key attributes.

The integrity constraint is mainly to avoid a user seeing null values for some of the elements forming the apparent primary key.

Polyinstantiation Integrity [2]. Given a relation R with the apparent primary key AK and key class C_{AK}, the following functional dependency holds for each attribute A_i which is not in AK:

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

In addition, R satisfies the following multivalued dependency for each attribute A_i,which is not in AK:

$$AK, C_{AK} \twoheadrightarrow A_i, C_i$$

The reason for that there must be the multivalued dependency property in the Sea View model is that instantiations are represented by distinct tuples if an element of an attribute A_i is polyinstantiated. Moreover, the multivalued dependency property means that for any two tuples with the same apparent primary key AK and key class C_{AK}, the values for any pair A_i, C_i can be swapped and the resulting tuple will also be in the relation R.

The problem of using this property to regulate polyinstantiation in a multilevel relation is that the number of manufactured tuples is exponential in the number of polyinstantiated elements. We explain the problem by the following example.

Loan-number	C_1	Customer-name	C_2	Amount	C_3	Interest-rate	C_4	TC
251105	TS	Smith	TS	15000	TS	825	TS	TS
105692	S	Adams	S	4500	S	850	S	S
141251	C	Glenn	C	2500	C	900	C	C

Figure 4. The Borrow-3 Relation: No Polyinstantiation

For example, consider the Borrow-3 relation in Figure 4 and suppose that an S user requires the following three operations:

UPDATE Borrow-3
SET Customer-name = "Hayes"
WHERE Loan-number = "141251"

UPDATE Borrow-3

SET Amount = 5200
WHERE Loan-number = "141251"

UPDATE Borrow-3
SET Interest-rate = 875
WHERE Loan-number = "141251"

After the three operations, the result is illustrated in Figure 5.

Loan-number	C_1	Customer-name	C_2	Amount	C_3	Interest-rate	C_4	TC
251105	TS	Smith	TS	15000	TS	825	TS	TS
105692	S	Adams	S	4500	S	850	S	S
141251	**C**	**Hayes**	**S**	**5200**	**S**	**875**	**S**	**S**
141251	C	Hayes	S	5200	S	900	C	S
141251	**C**	**Hayes**	**S**	**2500**	**C**	**875**	**S**	**S**
141251	C	Hayes	S	2500	C	900	C	S
141251	**C**	**Glenn**	**C**	**5200**	**S**	**875**	**S**	**S**
141251	C	Glenn	C	5200	S	900	C	S
141251	**C**	**Glenn**	**C**	**2500**	**C**	**875**	**S**	**S**
141251	C	Glenn	C	2500	C	900	C	C

Figure 5. The Borrow-3 Relation: Three Elements Are Polyinstantiated

Comparing Figure 4 and Figure 5, there are only three elements polyinstantiated but there exists eight tuples for the identical entity, Load-number = "141251". This creates a problem that the number of manufactured tuples is exponentially increased with the number of polyinstantiated elements.

Moreover, the existence of the cover story is also a problem after polyinstantiating an element because it conflicts with integrity. Considering Figure 5, the **Confidential** customer-name "Glenn" is a cover story for the real **Secret** customer-name "Hayes".

2.2 Jajodia-Sandhu Model

In the SeaView model, part of the tuples that result from the multivalued dependency property are not necessary from an intuitive point of view. For example, the 4,5,6,7,8,9th tuples of Figure 5 are **spurious** and can be left out when they are compared with the third tuple. Thus, a new polyinstantiation integrity constraint [10] proposed by Sandhu, Jajodia and Lunt requires that each entity in a multilevel relation has at most one tuple per security level. This constraint is stated as follows:

(PI-Tuple-Class) [10]. A multilevel relation R satisfies tuple-class polyinstantiation integrity if and only if for every instance in R_c,

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

In addition, the following two basic properties that are similar to the two properties, multilevel entity integrity and functional dependency, in the Sea View model must be satisfied in a multilevel relation R as well:

(Entity Integrity) [10]. Let AK be the apparent primary key of R. The instance set R_c of R satisfies entity integrity if and only if for all $t \in R_c$

1. $A_i \in AK \Rightarrow t[A_i] \neq$ null (Note the notation $\Rightarrow$ denotes logical implication)
2. $A_i, A_j \in AK \Rightarrow t[C_i] = t[C_j]$, that is, AK is uniformly classified.
3. $A_i \notin AK \Rightarrow t[C_i] \geq t[C_{AK}]$, where C_{AK} is the security level of the apparent primary key AK.

(PI-FD) [10]. A multilevel relation R satisfies functional dependency polyinstantiation integrity if and only if for every instance in R_c we have for all A_i

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

The property, **PI-FD**, prohibits polyinstantiation within a single security level. For example, it is not allowed that there are two values labeled U for the Objective attribute of the Enterprise entity in Figure 6.

Starship	C_1	Objective	C_2	Destination	C_3	TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Spying	U	Rigel	S	S

Figure 6. The SOD Relation

This property creates the update propagation problem due to data redundancy. If a user requires to update an element at the user's security level and there are duplications of this element in other tuples that represent the identical entity, the new value of the element must be propagated into other tuples with higher tuple-classes.

Loan-number	C_1	Customer-name	C_2	Amount	C_3	Interest-rate	C_4	TC
251105	TS	Smith	TS	15000	TS	825	TS	TS
105692	S	Adams	S	4500	S	850	S	S
141251	C	Hayes	S	5200	S	900	C	S
141251	C	Glenn	C	2500	C	900	C	C

Figure 7. The Borrow-3 Relation: One Tuple Per Tuple-class for the Identical Entity

For example, consider Figure 7 and suppose that a C user requires to update the Interest-rate of the entity "141251" to 875. After the update operation, the Interest-rate of the fourth tuple in Figure 7 is updated to 875 and the system must propagate the value, 875, into the third tuple, which is only contained in the S-instances or above and not visible to a C user.

3 Previous Polyinstantiation Elimination Methods

Because polyinstantiation in a multilevel relation results in the cover story, data redundancy and need of update propagation, it is desired to search a feasible method for eliminating entity polyinstantiation and element polyinstantiation. In this section, we review some previous work [7, 9, 11, 13].

3.1 Eliminating Entity Polyinstantiation

The major viewpoint for eliminating entity polyinstantiation is to prevent the same apparent primary key value from existing in a multilevel relation or database because entity polyinstantiation means tuples with the same apparent primary key value have different apparent primary key classes associated with the apparent primary key. The feasible solutions for eliminating entity polyinstantiation are explained as follows:

- Partition the global namespace into mutually exclusive namespaces at different security levels [7, 13]. This means that for a multilevel relation, a user is only permitted to insert a tuple whose apparent primary key value is classified at the user's security level.
- Make all keys visible [11] or prevent people with a low security level from assigning entity identifier (insert a new tuple) [7]. The method requires that in a multilevel relation, all apparent primary keys have the same security level and the security level is equal to that of the relation schema.

3.2 Eliminating Element Polyinstantiation

Sandhu and Jajodia use the value **restricted** to eliminate element polyinstantiation [9]. The approach requires that when any attribute of an entity has a non-null value at a given security level, the value of the attribute will be restricted at any security level which does not dominate the security level of the non-null value. A field with **restricted** value means that a user having the same classification as this field can no longer update it. To illustrate this approach, consider the relation in Figure 4 and suppose that an S user attempts to update the Amount of the entity "141251" to 5200.

Loan-number	C_1	Customer-name	C_2	Amount	C_3	Interest-rate	C_4	TC
251105	TS	Smith	TS	15000	TS	825	TS	TS
105692	S	Adams	S	4500	S	850	S	S
141251	C	Glenn	C	**restricted**	C	900	C	C

Figure 8. The Borrow-4 Relation: Mark an Element as Restricted

Step 1. The S user first logs in as a C subject and marks the Amount of the entity "141251" as **restricted**. The result is illustrated in Figure 8.

Step 2. The S user then logs in as an S subject and modifies the Amount of the entity "141251" to 5200. The result is illustrated in Figure 9.

Loan-number	C_1	Customer-name	C_2	Amount	C_3	Interest-rate	C_4	TC
251105	TS	Smith	TS	15000	TS	825	TS	TS
105692	S	Adams	S	4500	S	850	S	S
141251	C	Glenn	C	5200	S	900	C	S
141251	C	Glenn	C	restricted	C	900	C	C

Figure 9. The Borrow-4 Relation: Modify Amount to 5200

The order of Step 1 and Step 2 cannot be swapped because it will result in a temporary inconsistency in the multilevel relation.

Although this approach avoids cover stories in a multilevel relation, propagation of update is still needed. Besides, this approach is not suitable for the following two situations:

- Attributes of a multilevel relation have different security levels. For example, consider a salary relation and suppose that both the Employee-ID attribute and the Employee-name attribute are classified at the **Unclassified** security level and the Salary attribute is classified at the **Confidential** security level. If we use the **restricted** approach, the result is illustrated in Figure 10. There are two tuples for each employee in this salary relation.

Employee-ID	C_1	Employee-name	C_2	Salary	C_3	TC
00001	U	Jones	U	30000	C	C
00001	U	Jones	U	restricted	U	U
00002	U	Smith	U	26000	C	C
00002	U	Smith	U	restricted	U	U

Figure 10. The Salary Relation

- The existence of data with a higher security level cannot be inferred by a user at a lower security level. For example, consider the Crime attribute of the citizen relation in Figure 11. If a citizen has the value **restricted** at the Crime attribute, he or she will be thought to have at least one criminal record.

ID	C_1	Name	C_2	.....	Crime	C_n	TC
0000001	U	Jones	U	.....	null	U	U
:							
0012567	U	Curry	U		restricted	U	U
:							
0148652	U	Jones	U	.....	null	U	U

Figure 11. The Citizen Relation

4 Problems and Basic Ideas of Our Approach

Although the **restricted** element polyinstantiation elimination method proposed by Jajodia and Sandhu prevents cover stories existing in a multilevel relation, it does not avoid the need of update propagation (because there is data redundancy) and is only suitable for the situation where a user at a lower security level can be permitted to know the existence of data with a higher security level. Thus, we propose an improved element polyinstantiation elimination scheme which also solves the above two problems simultaneously. However, as analyzed in Section 5.3, this improved scheme also eliminates the entity polyinstantiation.

For the first problem, to avoid the need of update propagation, our scheme decomposes a multilevel relation into several subrelations where all attributes of each tuple are classified at the same security level. A multilevel relation is then regarded as a view over its subrelations. The following example as shown in [2] is used to explain why our decomposition method can avoid the need of update propagation.

Considering the multilevel relation R in Figure 12, the Figure 13 is single-level subrelations of R and the Figure 14 is views over these subrelations. The view to a **Secret** subject derives from unions and joins of all subrelations whose level is **Secret**. Now suppose that a **Secret** user requires to update the A_3 of Joe to become w. For the requirement, only one element in the **Secret** subrelation $R_{3,S,S}$* needs to be modified, but there are three elements in the two views of the Figure 14 to be changed to w.

A_1	C_1	A_2	C_2	A_3	C_3
Joe	S	45	S	y	S
Joe	S	45	S	x	TS
Joe	S	75	TS	y	S
Joe	S	75	TS	x	TS
Sam	TS	80	TS	z	TS

Figure 12. A Multilevel Relation R

$R_{1,S}$

A_1
Joe

Secret

$R_{2,S,S}$

A_1	A_2
Joe	45

Secret

$R_{3,S,S}$

A_1	A_3
Joe	y

Secret

$R_{1,TS}$

A_1
Sam

Top Secret

$R_{2,S,TS}$

A_1	A_2
Joe	75
Sam	80

Top Secret

$R_{3,S,TS}$

A_1	A_3
Joe	x
Sam	z

Top Secret

Figure 13. Subrelations of R

*The 1st subscript indicates which attribute of R the 2nd column of $R_{3,S,S}$ comes from. The 2nd subscript indicates the security level of the 1st column of $R_{3,S,S}$,and the 3rd subscript indicates the security level of the 2nd column of $R_{3,S,S}$.

A_1	C_1	A_2	C_2	A_3	C_3
Joe	S	45	S	y	S

A View to a **Secret** Subject

A_1	C_1	A_2	C_2	A_3	C_3
Joe	S	45	S	y	S
Joe	S	45	S	x	TS
Joe	S	75	TS	y	S
Joe	S	75	TS	x	TS
Sam	TS	80	TS	z	TS

A View to a **Top Secret** Subject

Figure 14. Views over these subrelations

Moreover, this scheme takes advantage of OID (object identifier) as a link among subrelations. There are two reasons:

1. With the same number of bytes, OID can usually represent more entities than the primary key attribute because part of key values may not be used in applications.
2. It avoids the occurrence of spurious tuples.

On the other hand, for the second problem, in a multilevel relation, to prevent a user at a lower security level from knowing existence of an object with a higher security level or inferring a rough value of an element with a higher security level, a feasible method is to make the attacked column only visible at a specific high classification (security level) or above, and all elements of the column to be uniformly classified at the specific high classification (that is, all elements of a column are either completely visible to a user or completely not visible to a user). Thus, in our scheme, when a user wants to create a multilevel relation in a database, we require the user to define two items, lowest-visible-classification(LVC) and whether-it-is-uniformly-classified(WIIUC), for each attribute. According to values of the two items, we decompose a multilevel relation into several subrelations where all attributes of each tuple have the same security level. For each user, the multilevel relation the user sees is a view which is the joins of all filtered subrelations whose schema security level is dominated by the security level of the user. When WIIUC of an attribute is N, it is permitted to let a user with a lower security level know the existence of data at a higher security level. Thus, the signaling channel can be ignored. When WIIUC of an attribute is Y, all elements are either completely visible to a user or completely not visible to a user. Thus, the signaling channel does not exist.

5 New Polyinstantiation Elimination Scheme

In this section, our scheme for eliminating

polyinstantiation is presented. The scheme decomposes a multilevel relation into subrelations according to LVC and WIIUC of each attribute, and uses OID (object identifier) to link among these subrelations. A set of constraints is also enforced in this scheme.

This section first describes how a multilevel relation is decomposed and then presents the set of constraints to be enforced in our scheme. Moreover, this section also analyzes why our scheme can eliminate polyinstantiation, avoid the need of update propagation and avoid updating a field from **restricted** to **unrestricted**.

5.1 Multilevel Relation Decomposition

To making our decomposition algorithm easily be understood, we first give an example of decomposition, and then describe our decomposition algorithm.

Decomposition example:

- Schema of a multilevel relation Loan:

begin schema
fieldname = Loan-number
type = C width = 6 LVC = C AK UNIQUE;
fieldname = Interest-rate
type = N width = 4 LVC = C WIIUC = Y;
fieldname = Amount
type = N width = 9 LVC = S WIIUC = Y;
fieldname = Customer-name
type = C width = 20 LVC = S WIIUC = N;
end schema

For each one of the data attributes composing the apparent primary key, its WIIUC is required to be Y. The main reason is to avoid entity polyinstantiation.

- Subrelations:

The subrelation Loan-1

schema classification = C uniform classification			
Loan-number	Interest-rate	OID	TC

The subrelation Loan-2

schema.classification = S uniform classification		
OID	Amount	TC

The subrelation Loan-3

schema classification = S non-uniform classification		
OID	Customer-name	TC

- Multilevel relation Loan for users at different classifications:

user's classification	multilevel relation Loan
U	deny
C	Loan-1
S	Loan-1 ⋈ Loan-2 ⋈ *filter*(Loan-3)
TS	Loan-1 ⋈ Loan-2 ⋈ Loan-3

For each tuple, the *filter* function is to replace the value of each attribute with **restricted** when the classification of the tuple is not dominated by that of the user. Moreover, the OID attribute is only used by the system, so the multilevel relation which a user sees does not usually include the OID attribute.

Decomposition algorithm:

Input:

Given a multilevel relation schema, and let P denote the set of all attributes in this schema.

Output:

subrelation schemas

Method:

Step 1. From the set P, select the attributes forming the apparent primary key (AK) and the attributes whose classification is equal to the classification of AK and its WIIUC is equal to Y.
Use these attributes with the OID attribute and a TC attribute to compose the first subrelation which is called main relation in our scheme.

Step 2. Select an attribute A_i with the lowest classification from the set P which does not include attributes that had been selected.
if WIIUC of the attribute is equal to N
then use the attribute, the OID attribute and a TC attribute to compose the next subrelation which is called attached relation in our scheme.
else from the set P, select other attributes whose classification is equal to the classification of A_i and WIIUC is equal to Y again.
Use the attribute A_i with other attributes, the OID attribute and a TC attribute to compose the next subrelation which is called attached relation in our scheme.

Step 3. *if* P is an empty set *then* goto Step 4 *else* goto Step 2.

Step 4. End.

5.2 Basic Constraints

The following basic integrity constraints are required in our scheme.

Property 1. **[Entity Integrity]** Let AK be the apparent primary key of a multilevel relation R and C_{AK} be the classification (security level) of the apparent primary key. R satisfies entity integrity if and only if
$\forall$ t $\in$ R such that
1. $\forall A_i \in AK \Rightarrow t[A_i] \neq null$.
2. $\forall A_i \in AK \Rightarrow t[C_i] = t[C_{AK}]$.
3. $\forall A_j \notin AK \Rightarrow t[C_j] \geq t[C_{AK}]$.

The property mainly avoids displaying AK which contains **null** values or **restricted** values to a user when the user wants to know which entity the value of

a specific attribute belongs to in a multilevel relation.

Property 2. **[Null Integrity]** A multilevel relation R satisfies null integrity if and only if the following conditions are both satisfied:

1. $\forall$ t $\in$ R's main relation, $A_i \notin AK$ and $t[A_i]$ = null $\Rightarrow$ $t[C_i]$= the classification of the main relation schema.
2. $\forall$ t $\in$ R's attached relations, $t[A_j]$ = null $\Rightarrow$ $t[C_j]$= the classification of the attached relation schema.

The property mainly avoids displaying the **restricted** value for an element to a user when the element with a higher classification was updated to null. In our scheme, the semantics of **null** is different from that of **restricted**. For an element, when the system displays **null** to a user, it permits the user to update the element. On the contrary, when the system displays **restricted** to a user, it does not permit the user to update the element.

Property 3. **[View Integrity]** A multilevel relation R satisfies view integrity if and only if for all C′ , the following requirement is satisfied:

For every tuple t in R's main relation or attached relations whose schema classification is dominated by the classification C′, when it is displayed to a C′ user , its view t′ to the C′ user is as follows:

$$t'[A_i] = \begin{cases} t[A_i] & \text{if } t[TC] \le C', \\ \text{restricted} & \text{otherwise.} \end{cases}$$

Subrelation Loan-1

Schema classification=C uniform classification			
L_N	I_R	OID	TC
121105	900	001	C
053692	950	002	C
004125	950	003	C
185429	900	004	C

Subrelation Loan-2

Schema classification=S uniform classification		
OID	Amount	TC
001	28000	S
002	15000	S
003	12000	S
004	65000	S

Subrelation Loan-3

Schema classification=S non-uniform classification		
OID	C_N	TC
001	Smith	S
002	Johnson	TS
003	Brooks	S
004	Greens	S

Figure 15. The Loan Relation Is Decomposed into Subrelations Loan-1, Loan-2 and Loan-3

For example, consider the relations in Figure 15 and suppose that an S user requires the following operation:

SELECT *
FROM Loan

The result is illustrated in Figure 16.

Loan-number	C_1	Interest-rate	C_2	Amount	C_3	Customer-name	C_4	TC
121105	C	900	C	28000	S	Smith	S	S
053692	C	950	C	15000	S	restricted	S	S
004125	C	950	C	12000	S	Brooks	S	S
185429	C	900	C	65000	S	Green	S	S

Figure 16. S-instances of the Loan Relation

Property 4. **[Element Integrity]** A multilevel relation R satisfies element integrity if and only if the following requirement is satisfied:

$$AK, C_{AK} \rightarrow A_i, \text{ where } A_i \notin AK$$

Different from the constrains (AK, C_{AK}, $C_i \rightarrow A_i$), the property constrains a multilevel relation R to have no element polyinstantiation.

Property 5. **[OID Reference Integrity]** A multilevel relation R satisfies OID reference integrity if and only if

1. For the OID value of each tuple in R's main relation, there exists the same OID value in each R's attached relation and it is unique in each R's attached relation.
2. If a tuple in R's main relation is deleted, then for each R's attached relation, the tuple whose OID value is identical to that of the deleted tuple in R's main relation must also be deleted.

Property 6. **[Tuple Deletion Integrity]** A multilevel relation R satisfies tuple deletion integrity if and only if for every tuple t $\in$ R, the following conditions hold if a C′ user is permitted to delete it:

1. The classification of its apparent primary key is C′.
2. There is no **restricted** in the maximal view of this tuple to the C′ user.

Under the interactive mode, a user can be permitted to delete a tuple only when at least the condition 2 of the property is satisfied. For example, considering Figure 15, both the C user and the S user can delete the third tuple of Figure 15. Under the non-interactive mode, a process is permitted to delete a tuple only when two conditions of the property are satisfied. This reason is to prevent a user at a lower classification from making use of a bit-leakage channel to obtain unauthorized information.

5.3 Analysis

In Sections 5.1 and 5.2, our scheme for eliminating polyinstantiation is introduced. This subsection explains why our scheme can eliminate both element polyinstantiation and entity polyinstantiation. Moreover, we explain why our scheme does not need to update an element from **restricted** to *unrestricted* or from *unrestricted* to **restricted**.

Our scheme decomposes a multilevel relation R into several subrelations according to whether the system, for an element in a column, can display **restricted** to a user with a lower security level if the element contains a data item with a higher security level (that is, whether a signaling channel can be ignored). For each of these subrelations, its data attributes are all either permitted or not permitted that

a signaling channel can be ignored. For a subrelation belonging to the latter, the signaling channel can not be ignored and the security levels of its elements are all the same. Therefore, polyinstantiation can not happen. On the other hand, for a subrelation belonging to the former, the signaling channel can be ignored and it uses the element integrity of our property 4 and the view integrity of our property 3 to eliminate polyinstantiation. For each element of the subrelation, the system decides to display either content of the element or **restricted** to a user according to the view integrity.

Furthermore, our scheme requires that all apparent primary keys in a multilevel relation have the same security levels. If the apparent primary key classes of entities (or tuples) are different, they will be stored in different relations. This requirement makes all keys visible and thus eliminates entity polyinstantiation.

In our scheme, although the system will display **restricted** for an element to a user of a lower security level according to the view integrity of our property 3, the element in its subrelation does not have to store a **restricted** value. For example, considering Figure 15 and Figure 16, the second tuple in Figure 15 which an S user sees contains a **restricted** value, but the corresponding field in Figure 16 is not a **restricted** value. Thus, our scheme does not need to update an element from **restricted** to *unrestricted* or from *unrestricted* to **restricted**.

In the worst case, using our decomposition method, a multilevel relation R with n attributes & s security levels might be divided into n*s subrelations. As R is divided into more subrelations, more join and union operations are needed in order to retrieve complete information of a tuple in R, which results in more query processing time. However, as compared with the decomposition method in [2], R in all cases, must be divided into n*s subrelations.

6. Conclusion

In the relational database with multilevel relations, there are two fundamental forms of polyinstantiation: entity polyinstantiation and element polyinstantiation. Entity polyinstantiation results in the cover story and data redundancy. Element polyinstantiation results in the cover story, data redundancy and the need of update propagation. In this paper, we propose a new scheme to eliminate both types of polyinstantiation as well as to resolve the cover story, data redundancy and the need of update propagation. As compared with the element polyinstantiation elimination scheme by Jajodia and Sandhu, this scheme avoids data redundancy, update propagation and the need of updating a field from **restricted** to *unrestricted* in a multilevel relation. Moreover, this scheme also makes the attribute name of a multilevel relation R with higher security level hidden from the user at lower security level.

References

1. S. Castano, M. G. Fugili, G. Martella, and P. Samarati, Database Security, Addison-Wesley, 1995.
2. D. E. Denning, T. F. Lunt, R. R. Schell, M. Heckman, and W. Shockley, "A multilevel relational data model," In Proc. IEEE Symp. on Security and Privacy, pp. 220-234, 1987.
3. D. E. Denning, T. F. Lunt, R. R. Schell, W. R. Shockley, and M. Heckman, "The SeaView security model," In Proc. IEEE Symp. on Security and Privacy, pp. 218-233, 1988.
4. S. Jajodia and R. S. Sandhu, "Toward a multilevel secure relational data model," Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 50-59, 1991.
5. S. Jajodia, R. Sandhu, and E. Sibley, "Update semantics for multilevel relations," Computer Security Applications Conf., pp. 103-115, 1990.
6. T. F. Lunt, "Aggregation and inference: facts and fallacies," In Proc. IEEE Symp. on Security and Privacy, pp. 102-109, 1989.
7. T. F. Lunt, "Polyinstantiation: an inevitable part of a multilevel world," Proc. The Computer Security Foundations Workshop IV, pp. 236-238, 1991.
8. J. K. Millen and T. F. Lunt, "Security for object-oriented database systems," In Proc. IEEE Symp. on Security and Privacy, pp. 260- 272, 1992.
9. R. S. Sandhu and S. Jajodia, "Eliminating polyinstantiation securely," Computers & Security, pp. 547-562, 1992.
10. R. S. Sandhu, S. Jajodia, and T. Lunt, "A new polyinstantiation integrity constraint for multilevel relations," Proc. The Computer Security Foundations Workshop III, pp. 159-165, 1990.
11. R. S. Sandhu and S. Jajodia, "Polyinstantiation for cover stories," Proc. Second European Symp. on Research in Computer Security, pp. 307-328, 1992.
12. G. W. Smith, "Identifying and representing the security semantics of an application," In Proc. of 4th Aerospace Computer Security Applications Conf., pp. 125-130, 1988.
13. M. B. Thuraisingham, "Mandatory security in object-oriented database system," In Proc. Conf. on Object-Oriented Programming: System, Language, and Application, pp. 203-210, 1989.
14. M. Winslett, K. Smith and X. Qian, "Formal query languages for secure relational database," ACM Transactions on Database Systems, Vol. 19 No. 4, pp. 626-662, 1994.