Polyinstantiation in Relational Databases with Multilevel Security

Andro Galinović
King ICT d.o.o., Buzinski prilaz 10, 10010 Zagreb, Croatia
E-mail: andro.galinovic@king-ict.hr

Vlatka Antončić
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
E-mail: vlatka.antoncic@vip.hr

Abstract. Polyinstantiation provides the ability to create more versions of single information. It is used to prevent inference attacks. This paper explains the use of polyinstantiation in relational databases with multilevel security for implementing e.g. cover stories. It describes common methods of access controls in relational databases and describes multilevel relational databases. This paper also shows how polyinstantiation can be implemented and what types of architecture support it.

Keywords. Database, security, relational, multilevel, polyinstantiation, discretionary access control, mandatory access control

1. Introduction

Database security and protection is often more complex problem then security and protection of other information systems. This is why security mechanisms that are embedded in most database management systems (DBMS) are quite complex. Because of the fact that databases contain structured and normalized data, information can be revealed indirectly or can be guessed, based upon some known information. There are two basic methods for revealing information, aggregation and inference.

Aggregation is a method used to collect less sensitive information or parts of them, from different sources which we have access to, in order to obtain aggregate information which is more sensitive.

Inference is a method with which we can deduce or infer the whole information, which we do not have access to, by observing available, less sensitive information.

Basic protection mechanisms are content-dependent access control, cell suppression, partitioning, noise and perturbation [10].

Most of these, and other methods, are based on denying information access to an individual. On one hand that is very efficient and ensures secrecy, but on the other hand it raises suspicions and confirms sensitivity of data. This presents a convert channel. The best way to convince that the information is not sensitive and that nothing is hidden is fake honesty. So, instead of denying access, cover stories are used.

Ability to create more versions of single information, depending on security level, is called polyinstantiation and is one of basic mechanisms in databases with multilevel security (MLS). The need for polyinstantiation was first noticed by T.H. Hinke and M. Schaefer in year 1975 [1]. The term polyinstantiation originated from SeaView project [3].

Most conflicts between security and integrity in multilevel secure DBMS come from incompatibility of basic principles of database security with integrity issues that have to be met [4]. Basic security principle is: security class of information should determine security class of all information it depends on.

2. Access control in relational databases

In DBMS discretionary access control (DAC) is usually implemented. In this kind of access control the owner of an object has discretionary right to grant access for an object to any subject. Two most common implementations of DAC are access control lists and access control matrix.

SQL’89 standard uses DAC and defines five types of privileges which establish the granularity of user access to the database: insert, delete, select, update and references. Privileges can be granted to one user or to all users (public). There is also a capability to grant access for
further access granting (with grant option). Standard does not define and does not demand possibility of revoking privileges. It is left up to the DBMS manufacturers.

SQL’92 standard added a possibility for more precise determination of privileges, possibility for determining activities in case of losing referential integrity and it added a command for revoking privileges (revoke).

The SQL:1999 standard contains support for non-discretionally access control model - role based access control (RBAC). Many manufacturers of DBMS have implemented some form of RBAC, e.g. IBM (Informix DBMS), Sybase and Oracle (9i - Oracle Label Security).

Mandatory access control (MAC) is more complex and secure model, which is based on the classification of information sensitivity and authorization level of subjects (users, programs ...) [11]. MAC is not directly supported in SQL or in relational databases. Polyninstantiation is often used in databases with MAC support to avoid inference [12].

3. Multilevel relational databases

According to [3], multilevel relation has polyninstantiation when it holds two or more tuples with the same primary key. In this context the primary key is the key that the database user defines as the primary. The true primary key that the user doesn’t see, and which is used by the DBMS while accessing tuples, is more complex and composite. This will be discussed in detail while explaining polyninstantiation integrity. The model that we will present in this section was developed by S. Jajodia and R Sandhu in 1991 [7].

Multilevel relation scheme R is defined as

\[ R(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, C_R) \]

where \( A_i \in D_i \) (that is, each attribute \( A_i \) is an element of its domain \( D_i \)), each \( C_i \) is classification of \( A_i \), and \( C_R \) is classification of the whole tuple. The domain of \( C_i \) is specified with a set \( \{L_i, \ldots, H_i\} \) which represents allowed values for access classes, ranging from \( L_i \) to \( H_i \). The domain of \( C_R \) is defined like a union of domains of each \( C_i \). An example of a classification can be: unclassified (U), classified (C), secret (S) and top secret (TS) [12].

There is one tuple \( t_c\{a_1, c_1, a_2, c_2, \ldots, a_n, c_n, c_R\} \) for each class with sensitivity \( c \), where \( a_i \in D_i \) or \( a_i = \text{null} \), \( c \geq c_i, c_i \neq \text{null} \), and \( c_i \) is maximum value of \( c_i \).

There are four basic integrity properties in multilevel databases that have to be fulfilled: entity integrity, null integrity, interinstance integrity and polyninstantiation integrity.

Entity integrity is defined as following. Let \( K \) be the apparent primary key (user defined) of a relation, and \( C_K \) classification of the key. If an attribute \( A_i \) is an element of the key \( (A_i \in K) \) it can not be null. This is an inherited property from the standard relational model. Secondly, for each two elements of the key \( (A_i, A_j \in K) \), classifications of that attributes have to be the same, that is \( t[C_i] = t[C_j] \). In other words the key is uniformly classified. This property insures that the entire key \( K \) is either visible or entirely null for the specific access class \( c \) (as we will see in interinstance integrity). Thirdly, for each element \( A_i \) that is not a part of the key \( (A_i \notin K) \), classification of that element must be the same or higher then the classification of the key \( (t[C_i] \geq t[C_K]) \). This property rules out the possibility of associating non-null attributes with a null primary key.

A multilevel relation \( R \) satisfies null integrity if and only if for each instance \( R_c \) of \( R \) both of the following conditions are true: null elements are classified with the same level as the key and relation \( t_c \) can not include two different tuples such that one subsumes the other.

Tuple \( t \) subsumes tuple \( s \) if for each attribute \( A_i \) either \( t[A_i, C_i] = s[A_i, C_i] \) or \( t[A_i] \neq \text{null} \) and \( s[A_i]=\text{null} \). Which means the tuple \( t \) is either identical to tuple \( s \) or that they differ only in those attributes in which one value is null.

Relation \( R \) satisfies interinstance integrity if and only if for all \( c' < c \) and \( R_{c'}=\sigma_{c'}(R_c) \) following statement is true: for each tuple \( t \in R_{c'} \) such that \( t[C_K] \leq c' \), exists a tuple \( t' \in R_c \) with \( t'[K, C_K] = t[K, C_K] \) and for \( A_i \notin K \nRightarrow t'[A_i, C_i] = \begin{cases} t[A_i, C_i], & \text{if } t[C_i] \leq c' \\ \{\text{null}, t[C_i]\}, & \text{otherwise} \end{cases} \)

Selection function \( (\sigma_{c'}(R)) \) returns one tuple for each access class. Function restricts each subject on reviewing only the part of multilevel relation for which it is authorized. This property
means that subject can only review information whose key has lower or equal level of classification than the subject itself. For all tuples that satisfy this condition, the subject can see the key and those values that have lower or equal level of classification than the subject. If the subject can’t access a certain value within the tuple he reads a null value. In this case null value can infer hidden data. This model also raises another question. When a query is executed with the classification c, and there are two tuples with classification c’ and c’’ respectively, where c > c’ > c’’, does the c classified subject reads the c’ or c’’ classified tuple? This is called query ambiguity.

Relation R satisfies polyinstantiation integrity if and only if for each tuple t A_i: K, C_K, C_i /g314 Ai applies. That is, for each tuple t apparent key K, classification of the key C_K, and classification of an attribute A_i (C_i) functionally determines the value of an attribute A_i.

Polyinstantiation integrity defines the primary key in a multilevel relation. The primary key of a multilevel relation is

\[ K \cup C_K \cup C_O \]

where K is a set of data attributes consisting the user specified primary key, C_K is a classification of that key, and C_O is a set of classification attributes for data attributes that are not part of the key (K)

\[ C_O= C_j, A_j \notin K \]

This property defines following function dependency

\[ K \cup C_K \cup C_O \rightarrow A_O \]

where A_O is a set of all attributes that are not part of the key.

This property enables the creation of cover stories i.e. tuples with the same user defined primary key but different data. This is used to protect against inference attacks.

The normal, one level relation, is a special case in which C_K = C_O for each tuple [7].

### 3.1. Types of multilevel relational databases

In multilevel relational databases access privileges can be defined on a relation level, tuple level, attribute level or data elements level. Polyinstantiation does not arise explicitly when access privileges are determined on relation level or attribute level. In this chapter we will discuss cases of access privileges that are defined on tuple level and on data element level.

When access privileges based on data element level are used, two types of polyinstantiation are possible: entity polyinstantiation and attribute polyinstantiation [6].

Entity (tuple) polyinstantiation appears when two or more tuples in one relation have the same apparent primary key but different key classifications.

Attribute (element) polyinstantiation appears when two or more tuples have the same apparent key and the same classifications of the key, but different values of one or more other attributes (that are not part of the key).

When access classification determined on tuple level is used, polyinstantiation means that more tuples, with the same apparent primary key and different data, based on classification of the tuple exist.

It is important to notice that if access classification is determined on tuple level, it is no longer possible to defer entity polyinstantiation and attribute polyinstantiation.

### 3.2. How the polyinstantiation occurs?

Polyinstantiation can occur in two possible ways. They are called visible and invisible polyinstantiation.

Visible polyinstantiation occurs when subject with higher level of clearance tries to insert information in a filed that already has that information, but with lower level of sensitivity. By changing that value, one of basic principles of mandatory model of access control would be violated. The *-property that says “don’t write down”[5]. That change would cause disclosure of information with higher classification to a subject with lower classification. In order to avoid this convert channel it’s necessary to create a new tuple for data with higher classification. This gives rise to polyinstantiation.

Invisible polyinstantiation occurs in reversed situation, when subject with lower level of classification tries to insert information in a filed that already has that information, but with higher level of sensitivity. If that change was to be declined, system would confirm existence of information with higher classification to a subject with lower classification. In order to avoid this convert channel it’s necessary to create a new tuple for data with higher classification. This gives rise to polyinstantiation.

Invisible polyinstantiation occurs in reversed situation, when subject with lower level of classification tries to insert information in a filed that already has that information, but with higher level of classification. If that change was to be declined, system would confirm existence of information with higher level of classification. In order to avoid this convert channel a new tuple is added for data with lower classification. This also gives rise to polyinstantiation.
4. Polyinstantiation example

Let us have three levels of classification, low (L), medium (M) and high (H). Let us take a look on multilevel relation scheme “missions”, with classification on attribute level, shown in Table 1. The apparent key of this relation are attributes Starship and Mission.

<table>
<thead>
<tr>
<th>Starship</th>
<th>Mission</th>
<th>Destination</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager L</td>
<td>Explore L</td>
<td>Nebula L</td>
<td>L</td>
</tr>
<tr>
<td>Voyager M</td>
<td>Lost M</td>
<td>Delta q. M</td>
<td>M</td>
</tr>
<tr>
<td>Enterprise L</td>
<td>On leave L</td>
<td>Risa L</td>
<td>L</td>
</tr>
<tr>
<td>Defiant L</td>
<td>Explore L</td>
<td>Wormhole M</td>
<td>M</td>
</tr>
<tr>
<td>Defiant H</td>
<td>Defense H</td>
<td>Dominion H</td>
<td>H</td>
</tr>
</tbody>
</table>

Table 1.

The subject with low (L) clearance can see data presented in Table 2.

<table>
<thead>
<tr>
<th>Starship</th>
<th>Mission</th>
<th>Destination</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager L</td>
<td>Explore L</td>
<td>Nebula L</td>
<td>L</td>
</tr>
<tr>
<td>Enterprise L</td>
<td>On leave L</td>
<td>Risa L</td>
<td>L</td>
</tr>
<tr>
<td>Defiant L</td>
<td>Explore L</td>
<td>Null L</td>
<td>L</td>
</tr>
</tbody>
</table>

Table 2.

The subject with medium (M) clearance can see data presented in Table 3.

<table>
<thead>
<tr>
<th>Starship</th>
<th>Mission</th>
<th>Destination</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voyager L</td>
<td>Explore L</td>
<td>Nebula L</td>
<td>L</td>
</tr>
<tr>
<td>Voyager M</td>
<td>Lost M</td>
<td>Delta q. M</td>
<td>M</td>
</tr>
<tr>
<td>Enterprise L</td>
<td>On leave L</td>
<td>Risa L</td>
<td>L</td>
</tr>
<tr>
<td>Defiant L</td>
<td>Explore L</td>
<td>Wormhole M</td>
<td>M</td>
</tr>
</tbody>
</table>

Table 3.

Only subject with high (H) clearance can see all data, as presented in Table 1.

5. Implementation of polyinstantiation

It is worth mentioning that MAC based DBMS can be designed without need of changing the SQL syntax. But, if true support for polyinstantiation is to be used, changes are needed. In most SQL based DBMS polyinstantiation can be implemented with built-in functionality and a few tricks [2]. A typical example of this is adding a view that would perform the selection function (σ_f(R)), mentioned earlier, and thus hide the complexity of a MLS relation.

There are several architectures used for implementing DBMS with mandatory access control. The most common are Trusted Computing Base (TCB) Subset Architecture, Trusted Subject Architecture and Integrity Lock Architecture [2]. All architectures depend on operating system so its confidentiality and reliability should not be questionable.

5.1. Trusted Computer Based Subset Architecture

Trusted Computer Based Subset Architecture implements MAC model by maintaining the database in multiple single-level files that are accessed through a trusted operating system that controls and determines access rights.

Architecture is suitable and with its design it implicates polyinstantiation. It enables data separation, based on classification, in different files, and access control is controlled by operating system based on DBMS instructions.

This architecture does not exclude the possibility of using a DAC model that is directly implemented in the DBMS.

The main advantage of this architecture is, that, it does not demand trust in DBMS, and the main flaw is that the number of files grows rapidly if we really want to support polyinstantiation.

Fig. 1 shows the scheme of this architecture.

5.2. Trusted Subject Architecture

Trusted Subject Architecture allows discretionary and mandatory access control. Data is stored in files of high classification of an operating system. In these files DBMS objects are labeled with their classifications. Those labels are basic for MAC model implementation.
This architecture does not implicate polyinstantiation, like TCB Subset Architecture, but it supports it. Support for polyinstantiation must be explicitly implemented if required. DAC model, just like in the case of the TCB Subset Architecture, is supported and based on the DBMS mechanisms.

Fig. 2 represents scheme of this model.

![Figure 2. Trusted Subject Architecture](image1)

5.3. Integrity Lock Architecture

Integrity lock architecture uses untrusted DBMS in combination with trusted operating system and trusted filter. DAC model, just like with previous models, is supported and based on DBMS mechanisms.

Trusted filter enables and implements MAC model and classification labeling. Cryptography and cryptography hash functions are used for securing classified data.

This architecture allows the usage of commercially available DBMS with additional security filters. The main disadvantage of this architecture is added overhead.

Fig. 3 represents scheme of this architecture.

![Figure 3. Integrity Lock Architecture](image2)

6. Analysis

To implement polyinstantiation, a record in the table has to be entered more than once. Each record will contain the same information, except for the classification level for which it is assigned.

Polyinstantiation contradicts basic database principals, like normalization and integrity. There is a tradeoff between security and integrity.

Polyinstantiation will also slow down the query response time, because of the extra processing required. The database has to search through multiple instances of the same record. Some of integrity constraints have to be sacrificed in order to increase security in database.

As stated earlier in section 3 this model has several drawbacks. First, it can infer hidden data through NULL values. Secondly, there is semantic ambiguity because tuples with the same apparent primary key but different key classification, can refer to different or the same information. Semantic ambiguity is addressed in the Smith-Winslett Belief model [9]. Thirdly, as already mentioned, ambiguity occurs during queries. One additional problem occurs, an update can result in an exponential increase in the number of tuples. This is called proliferation of tuples due to updates. A semantic solution to proliferation is addressed in Smith-Winslett Belief model [9]. In order to avoid wasting space it is possible to expand classification in a way that security label contains a list of classifications, and thus reduces the number of semantically identical tuples [13]. This principle is used in the Belief-Consistent Model [16][17].

Researchers continue to look for ways to resolve problems of inference. This problem is especially difficult in statistical databases [8][14][15].

7. Conclusion

Multilevel relational databases allow very complex security implementations and very precise access control. Polyinstantiation like a product of realization of multilevel security causes many dilemmas, over implementations themselves and over its need. Possibility for supporting polyinstantiation complicates systems and brings entropy, but if there is a need for that kind of precision in access control, multilevel databases polyinstantiation should be considered.
So far, the best way to defend from inference is the use of polyinstantiation. Until new methods are discovered it will be the best weapon against it. Curiosity is in human nature, and the best way to satisfy it, is through cover stories.

It is our opinion that polyinstantiation is not well suited for commercial use of databases, because commercial sector does not have so rigid security classifications defined.

This type of database security is more suited for military or government databases, where such structure already exists and where classified data needs to be protected.

8. References

[11] Lindqvist H. Mandatory Access Control, Masters thesis in computer science, Umea University, May 18, 22006