Querying a Multilevel Database: A Logical Analysis

Frédéric Cuppens
ONERA-CERT
2 Av. E. Belin
31055, Toulouse Cedex, France
email: cuppens@cert.fr
fax: +(33) 62 25 25 93

Abstract

This paper proposes a new formal model and semantics for answering queries in a multilevel polyinstantiated database. A polyinstantiated database may contain some contradictory facts distinguished by their classification. The main objective of this paper is then to provide the user cleared at a given level with a *consistent* and *complete* view of the multilevel database corresponding to his clearance level.

In our model, a multilevel database is simply viewed as a set of ordinary single-level databases. This model is based on propositional logic and each single-level database is associated with its logical models, as in the model-theoretic approach. It includes the possibility to hide some parts of the database schema. Finally, it may be used as a formal semantics for multilevel *deductive* databases.

Keywords: Multilevel Security Policy; Database Management; Propositional Logic; Model-theoretic Approach.

**Proceedings of the 22nd VLDB Conference
Mumbai(Bombay), India, 1996**

1 Introduction

An application that has been of particular interest since the beginning of work on secure computer systems is the implementation of a multilevel database management system (DBMS). Intuitively, a given multilevel policy assigns a clearance level to subjects and a classification level to sentences of the language used to represent the database content. Classifications and clearances are both taken in a set of *security levels.* For instance, the *sensitivity levels* Top Secret (TS), Secret (S), Confidential (C), and Unclassified (U) may be used as security levels. In this case, the set of sensitivity levels is associated with a total order, viz. $U < C < S < TS$. However, it is also possible to define a more complex set of security levels by combining a set of sensitivity levels with a set of *compartments.* Particular examples of compartments may be $NUCLEAR$, $MISSILE$... In this case, it is generally assumed that the security levels are partially ordered in a lattice. For instance, $(C, \{NUCLEAR\}) < (S, \{NUCLEAR, MISSILE\})$ because $C < S$ and $\{NUCLEAR\} \subset \{NUCLEAR, MISSILE\}$ but $(C, \{NUCLEAR\})$ and $(S, \{MISSILE\})$ are not comparable.

To design and construct a multilevel DBMS, we need a formal model. As a matter of fact, several security models for multilevel databases have been proposed in the literature. These models are generally based on the Bell-LaPadula model [BL75]. The Bell-LaPadula model imposes the two following requirements:

No read up Subjects are only permitted to read sentences stored in the database whose classification is lower or equal to their clearance.

No write down Subjects are only permitted to write sentences that are greater or equal to their clearance.

The "no write down" restriction is necessary to prevent a malicious agent (generally called a Trojan horse) from passing high classified data to a subject cleared at a lower level.

However, we argue that most of these proposals are not completely satisfactory, in particular, if the database may be polyinstantiated. In this case, the multilevel database may contain some contradictory facts distinguished by their classification. Many proposals exist in the literature to explain the need for polyinstantiation in a multilevel database. Our objective in this paper is not to investigate all these proposals but, in our view, the best explanation for polyinstantiation is that it may serve as a means to implement cover stories[1].

For example, let us consider the following multilevel instance of the relation scheme *Employee*(*Name*, *Salary*) where *Name* is the key:

Name	Salary	TC
Dupont	10000	C
Dupont	20000	S

The attribute *TC* represents the classification of the entire tuple. Since we assume that *Name* is the key of relation *employee*, *Dupont* may have only one salary and this relation is polyinstantiated. These contradictory tuples may be interpreted as follows: salary 20000 is the actual salary of Dupont. This is secret data and 10000 is a cover story, i.e. a lie provided to confidential users to hide the existence of a more secret salary 20000.

A problem which arises, in this case, is that the secret users are authorized to have a complete view of the database. They can observe both the secret data and the confidential cover story, the secret data being contradicted by the cover story. If the database provides the secret users with all these data without explanation, the secret users would be faced to an inconsistency.

In this paper, we propose a mechanism that enables the user cleared at a given level to be provided with a *consistent* and *complete* view of the multilevel database corresponding to his clearance level. This mechanism is based on the merging of the high level view of the database with the lower levels.

The remainder of this paper is organized as follows. Section 2 recalls the classical formalization of a non-protected database using propositional logic. Section 3 proposes a logical model for multilevel databases. It also formally defines what is a polyinstantiated database. Section 4 investigates how to answer a query in a multilevel polyinstantiated database. We actually compare three different approaches. The two first approaches are not completely satisfactory. The first one does not provide the user with a *consistent* view of the multilevel database. The second one does not provide the user with a *complete* view of the database corresponding to his clearance level. Therefore, we propose a third approach which enables these two problems to be solved. In section 5, we illustrate, through an example, how to use this last approach to provide answers to variously classified users. Section 6 is a discussion of our model. In section 7, we compare our approach with related work and section 8 concludes the paper on further work that remains to be done.

[1] A cover story is a lie which is provided to lower classified users to protect the existence of a higher classified data in the database

2 Non-protected database

The aim of this section is to fix a language to specify the content of a classical non-protected database. We assume that this language is based on propositional logic. Notice that, even if apparently a first order language seems to be required, we can actually assimilate it to a propositional one because of the domain closure assumption, i.e. we assume that there is a finite number of objects [Rei83]. In particular, a universally quantified formula may be, in this case, translated in a finite conjunction of formulae (one formula for each object) and an existentially quantified formula is translated in a finite disjunction. Therefore, the databases we consider here are represented by finite sets of formulae of a propositional language L.

Mathematical logic has been used to formalize databases, especially relational databases, in two main directions usually called the proof-theoretic approach and the model-theoretic approach. The former represents a database as a logical theory, the latter represents a database as an interpretation of a logical theory. In the remainder of this paper, we adopt the model-theoretic approach, i.e. each database is associated with its logical models. We shall use the following notations:

- $\models_m A$ is to be read: the interpretation m is a model of the propositional formula A.
- If DB is a set of propositional formulae, then v is a model of DB if and only if v is a model of every member of DB.
- DB^* denotes the set of all models of the set DB of formulae.
- $DB \models A$ is an abbreviation for $DB^* \subseteq \{A\}^*$, i.e. every model of DB is a model of A. $DB \models A$ is

to be read: the formula A is a logical consequence of the set DB of formulae.

We shall also assume that each database DB is a finite set of propositional formulae which are satisfiable (consistent), i.e. $DB^* \neq \emptyset$, but not necessarily complete, i.e. there may exist a formula A such that both A and $\neg A$ are not logical consequences of DB.

Let us now consider a user who wants to query the database DB. In our approach, a query is any formula of the language L and the answer to the query Q is defined as follows:

- $||Q|| =$ TRUE iff $DB \models Q$
- $||Q|| =$ FALSE iff $DB \models \neg Q$
- $||Q|| =$? iff else

Finally, we shall assume that there is a set of integrity constraints which describe the properties of the data stored in the database. The integrity constraints I are represented by a set of propositional formulae of L and we shall say that a database DB enforced the set of integrity constraints I if and only if $DB^* \subseteq I^*$ i.e. each formula of the set I is a logical consequence of the set DB.

3 Multilevel Database

We are now interested in databases which enforce the multilevel security policy. So, we have a finite set *Level* of security levels. We assume that the set *Level* is a lattice associated with a partial order relation denoted $<$. Therefore, the *least upper bound* and *greatest lower bound* are defined. For this purpose, we shall use two functions *lub* and *glb*. If l_1 and l_2 are two security levels, then $lub(l_1, l_2)$ and $glb(l_1, l_2)$ are respectively the least upper bound and greatest lower bound of l_1 and l_2. There is also a level which is lower than all other levels, we denote it $\perp$ and a level which is higher than all other levels, we denote it $\top$.

We also assume that the multilevel database is viewed as a set of single-level databases. This means that we partition the global multilevel database into single-level databases associated with each security level. This is formally represented as follows. A multilevel database DB is represented by a set of databases $\{DB_i, i \in Level\}$, each DB_i being a finite set of propositional formulae which are satisfiable but not necessarily complete. Moreover, each DB_i only contains formulae whose classifications are equal to i.

We shall also assume that there is a unique propositional language L upon which each database DB_i is defined, i.e. $DB_i \subseteq L$ for every $i \in Level$, and that there is unique set I of integrity constraints, each database DB_i enforcing this set I of constraints, i.e. $DB_i^* \subseteq I^*$. We agree that it may be interesting to consider that there is a different language L_i and a different set of integrity constraints I_i associated with each database DB_i. We shall not make these assumptions at this stage of the paper but we shall further discuss them in section 6.

Finally, we assume that the global multilevel database may be polyinstantiated. This is defined as follows: a multilevel database DB is polyinstantiated if and only if there is two security levels i and j such that $DB_i^* \cap DB_j^* = \emptyset$.

4 Defining consistent views

A user at a given security level is only permitted to observe all the single-level databases which are dominated by the user's clearance. Therefore, answering a query in a multilevel database depends on the level at which this query has been asked. For this purpose, we define the answer at level l to the query Q as follows:

- $||Q|| =$ TRUE iff $View\_At\_l \models Q$
- $||Q|| =$ FALSE iff $View\_At\_l \models \neg Q$
- $||Q|| =$? iff else

Intuitively, $View\_At\_l$ is the view of the multilevel database for users at level l. So the next question is how to define $View\_At\_l$?

If $l = \perp$, then it is easy to answer. $View\_At\_\perp$ is equal to the database $DB_\perp$, because users cleared at level $\perp$ are only permitted to observe the database $DB_\perp$. However, if l is strictly greater than $\perp$, it is not so easy to answer. We shall actually investigate three different possibilities. The two first possibilities are not completely satisfactory. We present them here because they are quite similar to previous attempts that have been proposed in the literature (see section 7 for a comparison).

4.1 Additive approach

The first approach, we call it the additive approach, consists in defining the view at a given level l as the union of all databases dominated by l, i.e. $View\_At\_l = \bigcup_{i \leq l} DB_i$.

It is easy to show that the set of models associated with $View\_At\_l$ is equal to the intersection of the sets of models associated with each DB_i i.e. $View\_At\_l^* = \bigcap_{i \leq l} DB_i^*$.

From a logical point of view, this approach is only correct if there is no polyinstantiation. As a matter of fact, if the multilevel database is polyinstantiated, then there exist two security levels i and j such that

$DB_i^* \cap DB_j^* = \emptyset$. Therefore, for every level k such that $k \geq lub(i,j)$, $View\_At\_k$ is an inconsistent theory. From a logical point of view, this has tragic consequences because we can actually derive any formula from an inconsistent theory.

Hence, we cannot consider that this approach is correct in the context of a polyinstantiated database.

4.2 Suspicious approach

The second approach is called the suspicious approach. In this case, the view at a given level l is simply equal to the database at level l, i.e. $View\_At\_l = DB_l$.

Since we assume that each DB_l is a consistent set of formulae, each user is provided with a consistent view of the multilevel database. Therefore, this approach is correct from a logical point of view.

However, the multilevel database looks like a set of single-level unrelated databases. There is no low data which are believed at a high security level. Our point of view is that we cannot consider that this is really a multilevel approach. It is rather a "system high" approach in which users cleared at a high security level have only access to high classified data.

4.3 Trusted approach

Faced to these difficulties, we propose a third approach called the trusted approach. We first notice that data stored in each single-level database generally correspond to a partial view of the universe by users at the corresponding security level. This is because we assumed that each single-level database DB_l only contains data classified at level l.

Therefore, in the trusted approach, the view at a given level l is obtained by merging the single-level database at level l with all the lower single-level databases. Intuitively, if a database at level l_{k-1} is consistent with a database at level l_k, then $DB_{l_{k-1}}$ can completely flow to level l_k (as in the additive approach). On the other hand, if $DB_{l_{k-1}}$ contradicts DB_{l_k}, then the trusted approach retrieves the largest subset of $DB_{l_{k-1}}$ which is consistent with DB_{l_k}. This subset can then flow to level l_k.

To formally represent this mechanism, we shall take our inspiration in the work of Katsuno and Mendelson [KM88]. Notice that, in this section, we assume that the set $Level$ is a set of p levels associated with a total order, i.e. $Level = \{l_1, l_2, ..., l_p\}$ with $l_1 \leq l_2 \leq ... \leq l_p$. The case of a partial order relation will be discussed in section 6. We then recall some definitions:

- Let L be a propositional language whose propositional variables is equal to $\mathcal{P} = \{p_1, p_2, ..., p_L\}$. Let m_1 and m_2 be two interpretations of L. We define the distance between m_1 and m_2 as follows:

$$d(m_1, m_2) = \{p \in \mathcal{P} :(\models_{m_1} p \text{ and } \models_{m_2} \neg p) \text{ or } (\models_{m_1} \neg p \text{ and } \models_{m_2} p)\}$$

- Let m be an interpretation. We define a partial order relation $\preceq_m$ on the set of interpretations as follows:

$$m_1 \preceq_m m_2 \longleftrightarrow d(m, m_1) \subseteq d(m, m_2)$$

- Finally, let M_1 and M_2 be two sets of interpretations. We define the fusion of M_1 with M_2, denoted $M_1 \rhd M_2$, as follows:

$$M_1 \rhd M_2 = \bigcup_{m \in M_1} Min(M_2, \preceq_m)$$

  Intuitively, $M_1 \rhd M_2$ is the subset of interpretations of M_1 which are the closest from the interpretations of M_2 according to the partial order relation $\preceq_m$.

Using these definitions, the set of models of $View\_At\_l_k$ is recursively defined as follows:

- $(View\_At\_l_1)^* = DB_{l_1}^*$
- $(View\_At\_l_k)^* = DB_{l_k}^* \rhd (View\_At\_l_{k-1})^*$

We can then prove the following theorems, for every $l \in Level$:

- $DB_l^* \neq \emptyset \rightarrow (View\_At\_l)^* \neq \emptyset$
- $(View\_At\_l)^* \subseteq DB_l^*$

The first theorem says that if DB_l is a consistent set of formulae, then $View\_At\_l$ is also a consistent set of formulae. Since we assumed that each DB_l is a consistent set of formulae, this ensures that each user is provided with a consistent view of the multilevel database.

The second theorem says that the models of $View\_At\_l$ are included in the models of DB_l. This implies that if A is a logical consequence of DB_l then A is also a logical consequence of $View\_At\_l$.

5 Example

This section proposes an example to illustrate the trusted approach. To describe this example, we use the concepts introduced in the Entity-Relationship model. This does not mean that our approach is restricted to this model. It may apply to the relational model as well. However, we prefer to use the Entity-Relationship model because several features are easier to explain using this model. Notice also that, for the sake of simplicity, our example is presented using a predicate-like notation but, as noticed in section 2, a propositional language would be actually sufficient.

In our logical approach, we first need a set of constants and a set of unary predicates to respectively identify the entities and to represent the class of an entity. For instance, the data "the entity whose identifier is o_1 is an instance of the class Employee" is represented in our logical theory by the following fact:

- $Emp(o_1)$

The data related to an entity are represented by a set of attributes. Hence, the set of constants must include a set of attribute values and the set of predicates must include a set of attribute predicates. For instance, we shall assume that an employee is associated with three attributes *Name*, *Age* and *Job*. Therefore, to represent an employee in our logic, we shall use three predicates: $Name(x,y)$, $Age(x,y)$ and $Job(x,y)$.

This language is sufficient for the purpose of our example. We refer to [CD89] for a more detailed presentation which includes the concept of relations, complex entity values and inheritance. It is somewhat straightforward to specify all these concepts in logic.

In our example, we also consider a set I of three integrity constraints:

- $\forall x \forall y_1 \forall y_2,$
  $Name(x,y_1) \wedge Name(x,y_2) \rightarrow y_1 = y_2$

- $\forall x \forall y_1 \forall y_2,$
  $Age(x,y_1) \wedge Age(x,y_2) \rightarrow y_1 = y_2$

- $\forall x \forall y_1 \forall y_2 \forall y_3,$
  $Job(x,y_1) \wedge Job(x,y_2) \wedge Job(x,y_3)$
  $\rightarrow y_1 = y_2 \vee y_1 = y_3$

The two first constraints say that the *Name* and *Age* of any entity are unique. The third constraint says that any entity cannot have more than two different jobs.

Finally, we use this language to represent our example of multilevel database (see figure 1). We assume in this example that there are only two security levels: $U = \perp$ (Unclassified) and $S = \top$ (Secret). This multilevel database is polyinstantiated. First, because the employee o_1 has two different ages, one unclassified and one secret. This is inconsistent with the second integrity constraint. Second, because the employee o_1 has three different jobs, two unclassified and one secret. This is inconsistent with the third integrity constraint. Let us also add, in the unclassified database, the following completion axiom for the predicate *Emp* [Rei83]:

- $\forall x, Emp(x) \rightarrow x = o_1$

By combining this axiom with the three above integrity constraints, we can derive that there is only one model of DB_U. If we only mention the propositions whose valuations are equal to true, this is represented as follows:

- $DB_U^* = \{[Emp(o_1), Name(o_1, Dupont),$
  $Age(o_1, 30), Job(o_1, Professor),$
  $Job(o_1, Engineer)]\}$

As we have $View\_At\_U^* = DB_U^*$, this also represents the view of the multilevel database at the unclassified level. On the other hand, there are many models for DB_S. However, if we evaluate $DB_S^* \rhd DB_U^*$, then we only obtain the two following interpretations for $View\_At\_S$:

- $View\_At\_S^* = \{[Emp(o_1), Name(o_1, Dupont),$
  $Age(o_1, 35), Job(o_1, Engineer),$
  $Job(o_1, Secret\_Agent)]$
  $[Emp(o_1), Name(o_1, Dupont),$
  $Age(o_1, 35), Job(o_1, Professor),$
  $Job(o_1, Secret\_Agent)]\}$

The fact that we obtain two different models for $View\_At\_S$ means that $View\_At\_S$ is incomplete (see figure 2). $Secret\_Agent$ is stored in $View\_At\_S$ because this is a secret job. On the other hand, both *Professor* and *Engineer* cannot be stored in $View\_At\_S$ because o_1 cannot have three jobs. In this case, the trusted attitude does not choose and a disjunctive fact $Job(o_1, Professor) \vee Job(o_1, Engineer)$ is generated.

Therefore, if a secret user queries the database to know whether Dupont is a professor, then the database will answer MAYBE if the trusted approach is used. We argue that this is a better choice than to answer YES as in the additive approach, or NO as in the suspicious approach (if the closed world assumption is done).

However, if disjunctive facts are to be avoided (for instance, because it is an implementation requirement), then the only way to proceed is to explicitly complete DB_S by inserting a positive fact, for instance $Job(o_1, Professor)$ or a negative fact, for instance $\neg Job(o_1, Engineer)$ (if the database provides means to manage negative data).

6 Discussion

6.1 Axiomatics

In the previous sections, we followed a model-theoretic approach and we propose a formal semantics for multilevel databases. In the present section, we address the problem of defining an axiomatics for the proof-theoretic approach. Actually, we have no complete

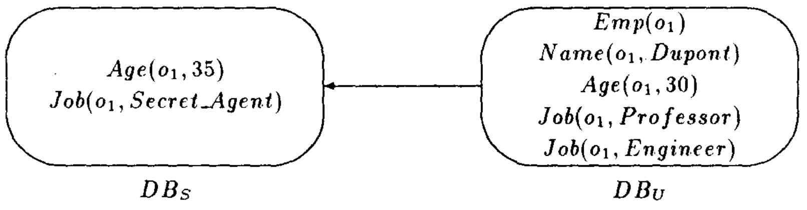

Figure 1: Unclassified and secret databases

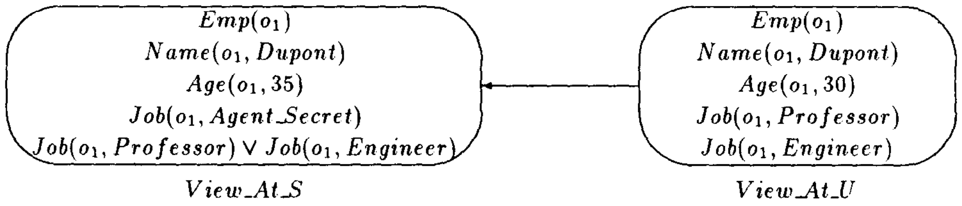

Figure 2: Unclassified and secret views of the multilevel database

axiomatics to propose. As a matter of fact, it is trivial to prove that the following rule is sound for every $l \in Level$:

- $\dfrac{DB_l \vdash A}{View\_At\_l \vdash A}$

i.e. if A is derivable from DB_l then A is also derivable from $View\_At\_l$.

However, it is much more difficult to derive which data stored in a lower classified database can flow in a higher classified view. For instance, we may think that the following rule is sound:

- $\dfrac{View\_At\_l_{k-1} \vdash A \quad DB_{l_k} \not\vdash \neg A}{View\_At\_l_k \vdash A}$

i.e. if A is derivable from $View\_At\_l_{k-1}$ and if $\neg A$ is not derivable from DB_{l_k} then A should be derivable from $View\_At\_l_k$. However, this rule is not sound. The example of section 5 provides a counter example. We have $View\_At\_U \vdash Job(o_1, Professor)$ and $DB_S \not\vdash \neg Job(o_1, Professor)$. So, by applying the above rule we could derive that $View\_At\_S \vdash Job(o_1, Professor)$. Similarly, we have $View\_At\_U \vdash Job(o_1, Engineer)$ and $DB_S \not\vdash \neg Job(o_1, Engineer)$, so we could also derive that $View\_At\_S \vdash Job(o_1, Engineer)$. On the other hand, by applying the first rule, we can derive $View\_At\_S \vdash Job(o_1, Secret\_Agent)$. Therefore, $View\_At\_S$ is inconsistent since o_1 has now three different jobs *Professor*, *Engineer* and *Secret_Agent*.

However, we guess that our second rule would be sound if we syntactically restrict the type of clauses to be stored in the database. In particular, we conjecture that this rule is sound if we only deal with definite clause, i.e. clauses of the form $A_1 \wedge ... \wedge A_n \rightarrow B$ in which all variables are assumed to be universally quantified. Proving this conjecture represents future work that remains to be done.

6.2 Schema protection

In section 3, we assumed that there is a single propositional language which is common to every single-level database $DB_i, i \in Level$. This implicitly means that we assumed that the schema of the multilevel database is classified at level $\perp$. This is an unnecessary restriction which can easily be removed. We have simply to consider that each single-level database DB_i is associated with its own language L_i.

Let us also consider that we partition the set of propositional variables into subsets associated with a security level, i.e. each single-level database DB_i is associated with a set $\mathcal{P}_i = \{p_{i_1}, p_{i_2}, ..., p_{i_k}\}$ of propositional variables. In this case, L_i can be defined by using the additive approach, i.e. L_i is a propositional language whose propositional variables is equal to $\bigcup\limits_{l \leq i} \mathcal{P}_l$.

Intuitively, $p \in \mathcal{P}_i$ means that p is a proposition classified at level i. This enables some part of the database schema to be protected. For instance, in the example of section 5, we can associate the class *Employee* with an additional attribute *Religion*. Let us consider that every atomic formula having the form $Religion(o, r)$ are in $\mathcal{P}_S$ but not in $\mathcal{P}_U$. This means that not only the fact that an employee o has a given religion r is classified at secret, but even the fact that the multilevel database manages this kind of data is classified at secret.

Notice that the definition of $View\_At\_l$ must be slightly modified. We first need to use the following notation:

- If DB is a set of formulae of language L then $(DB^*)_L$ represents the set of models of DB in language L.

DB_i is a subset of formulae of language L_i. However, from the above definitions, it is also a subset of formulae of language L_j if $j \geq i$. Therefore, using this notation, the set of models of $View\_At\_l_k$ is recursively defined as follows:

- $(View\_At\_l_1^*)_{L_{l_k}} = (DB_{l_1}^*)_{L_{l_k}}$
- $(View\_At\_l_k^*)_{L_{l_k}} = (DB_{l_k}^*)_{L_{l_k}} \triangleright (View\_At\_l_{k-1}^*)_{L_{l_k}}$

However, in the remainder of this paper, we shall omit to precise the language of definition when no confusion is possible.

6.3 Integrity constraints

In section 3, we also assumed that there is a single set of integrity constraints which is common to every single-level database $DB_i, i \in Level$. This means that we assumed that the integrity constraints are classified at level $\bot$. To remove this restriction, We have to consider that we partition the global set of integrity constraints into subsets I_i associated with each single-level database DB_i.

For instance, let us assume that the following integrity constraint i_1 is stored at the unclassified level:

- $\forall x, \forall y, Emp(x) \wedge Age(x,y) \rightarrow y \leq 65$

i.e. an employee must stop working after 65 years.

On the other hand, let us assume that there are employees who may continue working until 70 years but this data must be kept secret. Therefore, we can state the following integrity constraint i_2 at the secret level:

- $\forall x, \forall y, Emp(x) \wedge Age(x,y) \rightarrow y \leq 70$

As two different sets of integrity constraints I_i and I_j may be conflicting, i.e. $I_i^* \cap I_j^* = \emptyset$, we might suggest using the trusted approach to derive the set of integrity constraints $Integrity\_At\_l_k$ which apply to the security level l_k:

- $(Integrity\_At\_l_1)^* = I_{l_1}^*$
- $(Integrity\_At\_l_k)^* = I_{l_k}^* \triangleright (Integrity\_At\_l_{k-1})^*$

However, this definition has the following drawback. In the above example, i_2 is "weaker" than i_1 in the sense that $\{i_1\}^* \subseteq \{i_2\}^*$. Therefore, when evaluating $Integrity\_At\_S$, the unclassified constraint i_1 flows to the secret level and overrides the secret constraint i_2. This is likely an undesirable effect because it would no longer be possible to insert at the secret level an employee whose age is greater than 65. Solving this problem also represents future work that remains to be done.

6.4 Case of a partial order

Finally, in section 4, we assumed that *Level* is associated with a total order. Let us now investigate the case of a partial order. In this case, it is more difficult to define what is the actual complete view at a given level.

For instance, let us consider four security levels U, $C1$, $C2$ and S with $U \leq C1 \leq S$ and $U \leq C2 \leq S$. $C1$ and $C2$ are not comparable. Let us now assume that $Age(o_1, 30)$ is stored in database DB_{C_1}, $Age(o_1, 35)$ is stored in DB_{C_2} and there is not explicit age stored in DB_S. In this case, how to derive $View\_At\_S$? A possible solution would be to evaluate $View\_At\_S$ as follows:

- $View\_At\_S^* = DB_S^* \triangleright (DB_{C_1}^* \bowtie DB_{C_2}^*) \triangleright DB_U^*$

with $(DB_{C_1}^* \bowtie DB_{C_2}^*)$ defined by:

- $(DB_{C_1}^* \bowtie DB_{C_2}^*) = (DB_{C_1}^* \triangleright DB_{C_2}^*) \cup (DB_{C_2}^* \triangleright DB_{C_1}^*)$

i.e. DB_{C_1} and DB_{C_2} are merged by solving the possible conflicts but without specifying a preference in case of conflict. In this case, the disjunctive fact $Age(o_1, 30) \vee Age(o_1, 35)$ would be generated in $View\_At\_S$.

Another more elaborated solution proposed in [CC95] would be to use the concept of topic (see [CD89]) to determine which age is to be preferred. However, due to space limitation, we cannot develop this approach in the present paper.

7 Comparison with related work

In this section, we consider the concepts that have been proposed in other multilevel database models, mainly in the context of relational databases, and make observations and comparisons with our own.

7.1 The Seaview Model

The Seaview model [DLS+87, DLS+88] is a joint project between SRI International and Gemini Computers to product a multilevel relational database. This project uses attribute level security, i.e. each non-protected relation $R(A_1, ..., A_n)$ becomes a multilevel relation:

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

where each A_i is a data attribute over a domain D_i, each C_i is a classification attribute for A_i and TC is the class attribute. A relation instance is a set of tuples, each of the form:

$$(a_1, c_1, a_2, c_2, ..., a_n, c_n, tc)$$

where each $a_i \in D_i$, $c_i \in Level$ and $tc = lub(c_1, c_2, ..., c_n)$. At first sight, it does not seem possible to represent, in this case, a multilevel database by a collection of single-level databases, as was suggested in section 3. However, in Seaview, each multilevel relation are decomposed into a collection of single-level relations:

- $R_1(A_K, C_K)$ where A_K are the key attributes of the non-protected database R and C_K the key classifications[2].
- $R_i(A_K, C_K, A_i, C_i)$ for each non-key attribute A_i.

Each of these relations is then stored into single-level databases. This is fully compatible with the model proposed in section 3. In Seaview, there is also a recovery algorithm which provides a user cleared at a given level with a view of the database at the corresponding security level. The central idea of this recovery algorithm is close to the additive approach.

On the other hand, Seaview allows the multilevel database to be polyinstantiated. Therefore, as was told in section 4, the user is provided with a view which is globally inconsistent. We guess this may lead to undesirable consequences.

However, the reader may believe that problems are avoided if the user sees the classification of the retrieved facts. This is not always true, especially in the case of a conjunctive query. For instance, let us consider a secret user who asks the following query to the database of section 5:

- $Q = Age(o_1, 30) \wedge Job(o_1, Secret\_Agent)$

Using the additive approach, the answer to this query is *True* and the multilevel database can specify that this answer is secret since $Job(o_1, Secret\_Agent)$ is stored in the secret database. In this case, this user will believe that the age of o_1 is 30 without being aware that $Age(o_1, 30)$ is an unclassified fact.

So, if this user asks another query, now replacing 30 by 35:

- $Q = Age(o_1, 35) \wedge Job(o_1, Secret\_Agent)$

he will also obtain *True* as a secret answer and he will now implicitly assume that the age of o_1 is 35. So, the additive approach has the bad effect that this user may believe that the database is inconsistent.

[2]In Seaview, the key attributes are constrained to be uniformly classified

7.2 The Jajodia-Sandhu model

The Jajodia-Sandhu model [SJ92] is quite similar to the Seaview model. However, the polyinstantiation problem is more accurately investigated. As was first suggested by Lunt in [Lun91], Jajodia and Sandhu propose to distinguish between two types of polyinstantiation: entity polyinstantiation and element polyinstantiation.

7.2.1 Entity polyinstantiation

Entity polyinstantiation may occur when a relation contains multiple tuples with the same apparent primary key values, but having different classification for this apparent primary key. For instance:

Employee		Name		Age		TC
o_1	U	Dupont	U	30	U	U
o_1	S	Martin	S	35	S	S

A possible interpretation, suggested by [Lun91], would be to consider there are actually two distinct entities in the external world respectively identified by the pairs (o_1, U) and (o_1, S). However, this would be against the philosophy of the Entity-Relationship model where the identifier is usually used to uniquely identify a real world entity. Therefore, we guess that it is generally better to prevent entity polyinstantiation. [SJ92] suggest that there are several techniques for eliminating entity polyinstantiation. One possibility is to partition the domain of the entity identifiers. For instance, one can specify that the symbols beginning with "S-" are secret symbol. In this case, the multilevel database can reject any attempt by an unclassified user to insert an entity whose name begins with "S-". This enables entity polyinstantiation to be prevented.

7.2.2 Element polyinstantiation

Element polyinstantiation occurs when a relation contains two or more tuples with identical apparent primary keys and associated key classification, but having different values for one or more remaining attributes. For instance:

Employee		Name		Age		TC
o_1	U	Dupont	U	30	U	U
o_1	U	Dupont	U	35	S	S

A possible interpretation of element polyinstantiation is the following. The age 30 may be viewed as a cover story, i.e a lie provided to unclassified users to hide the existence of the more secret age 35.

Element polyinstantiation is a special case of polyinstantiation as defined in section 3. However, our definition is more general since it enables to consider that the attribute *Job* is polyinstantiated in the example of section 5 whereas it does not correspond to an element polyinstantiation.

7.3 The Smith-Winslett model

The formal model proposed by Smith and Winslett [SW92] is quite similar to our own. Under their semantics, the interpretation of a multilevel database is a set of ordinary relational databases. Smith and Winslett also investigate how to query a multilevel database. Our suspicious approach is actually directly derived from their approach. A user cleared at a given level only believes in the data stored in the single-level database corresponding to his clearance and does not believe in data stored in lower classified databases. From a logical point of view, this assumption is correct but too restrictive to be considered a multilevel approach.

7.4 The Spalka model

Spalka [Spa94] also proposed a model for multilevel databases based on mathematical logic. In his approach, a multilevel database is represented by a single theory. A user cleared at a given security level is then provided with a partial view of this global theory compatible with his clearance. This view also depends on which formal definition of confidentiality is chosen. Spalka investigates four different possibilities which correspond to the informal answer ***Maybe, No, Don't know*** and ***Don't understand***. These four definitions capture the various degrees of implicit information a user may obtain on a secret.

[Spa94] is an interesting proposal. However, we guess that it is a conceptual simplification to represent a multilevel database as a set of single-level theories instead of a single multilevel theory. In particular, we guess that a single multilevel theory would be more difficult to manage and to update without using a centralized authority which controls every update. Moreover, in Spalka's approach, there are no polyinstantiated facts stored in the multilevel database. This is an indirect consequence of the fact that Spalka considers only one global theory because this global theory would be inconsistent if it contained polyinstantiated facts. Therefore, if the chosen definition of confidentiality requires to provide the user with a cover story, then a possible solution would be that the database management system automatically generates this cover story. However, we guess that this objective would be difficult to achieve because a cover story generally requires a large amount of knowledge to be properly generated.

7.5 The Thuraisingham model

There also exists some connection between our approach and the Nonmonotonic Typed Multilevel Logic (NTML) developed by Thuraisingham [Thu91]. However, we prefer representing each single-level database as a set of models instead of as a theory. This is because, in many specific situations it would not be clear how to derive a consistent view using NTML. These situations include the case of a partial ordering of the security levels but are not restricted to this case. In particular, NTML is faced to the same problem as the one mentioned in section 6.1. For instance if P and Q are facts at the unclassified level and $\neg(P \wedge Q)$ a new fact at the secret level, it is not clear how to avoid a contradiction at the secret level using NTML. This problem was first noticed in [GLQS92].

7.6 The MultiView model

The MultiView model [BCC93, BCC94] is a model for multilevel object oriented databases. MultiView also represents a Multilevel database as a collection of single-level databases but each single-level database represents a *complete* view of the universe at the corresponding security level. For this purpose, MultiView suggests using dynamic links between each single level (see figure 3). In this figure, (O_1, U) and (O_1, S) respectively represents the view at the unclassified and secret level of a unique object O_1. The value of the attribute *Name* in the secret view (O_1, S) is equal to $(O_1, U).Name$ which stands for a dynamic link to the name value of (O_1, U). Intuitively, this dynamic link avoids replication of the same information at several security levels and makes automatic the propagation of a low level update to the higher levels.

Using the notations of our paper, MultiView dynamically manages, for each $l \in Level$, a database corresponding to $View\_At\_l$ instead of DB_l. Therefore, MultiView may provide a concrete implementation of the model describes in this paper in the special case of a database restricted to be a set of atomic facts.

8 Conclusion

When using polyinstantiation in a multilevel database, several problems are to be solved. This paper aims to solve one of these problems, namely how to provide the user with a consistent view of the multilevel database. We have proposed a formal model and semantics for answering multilevel queries in this case. This model includes the possibility to hide some parts of the database schema and to deal with rules in the database. Therefore, it may also be used as a formal semantics for multilevel *deductive* databases.

There are several issues to this work. In section 6,

Object: (O_1 S)
Name: (O_1 U).Name
Age: 35

Object: (O_1 U)
Name: Dupont
Age: 30

Figure 3: Example of MultiView database

we have already mentioned that a complete axiomatics for this model is to be investigated. Moreover, even though section 7.6 suggests some preliminary ideas, the general problem of how to efficiently answer queries within the proposed model is not addressed in this paper. How to deal with multilevel integrity constraints also requires further investigations. Another refinement would be to include the case where we do not need polyinstantiation, i.e. we do not want to hide the existence of a secret event. For this purpose, we can use the special symbol *Restricted* introduced in [SJ91]. Intuitively, an attribute assigned to *Restricted* means that the value exists but is higher classified. We do not feel that it would generate any problem because, in this case, the high view is not contradicted by the lower view.

Our approach is designed to provide the high level user with some parts of the low level database which are not in contradiction with the high level database. It would also be interesting to extend this approach so that it would be possible for the high level user to know if the lower users are believing some cover stories. As our approach is designed to recognize if a given data is a cover story, we guess that it would probably be easy to include this extension in our model.

Another problem not addressed in this paper is how to properly update a multilevel database. This problem has already been discussed before, for instance by Smith and Winslett [SW92] or by Sandhu and Jajodia [JSS90]. However, a problem generally not addressed is how to propagate a low level update to higher levels. For instance, [SW92] assumes that an update performed by a user at a given level can only alter data in the interpretation at the subject's own level. This is clearly the simplest solution. However, let us assume that a low level cover story is updated. Have we to consider that the new data stands for a new cover story (in which case no propagation is to be done) or as the new reality (in which case higher data is also to be updated)? We guess that further work are necessary to provide a clear answer to this question.

References

[BCC93] N. Boulahia-Cuppens, F. Cuppens, A. Gabillon, and K. Yazdanian. Multi-View Model for Multilevel Object Oriented Databases. In *Ninth Annual Computer Security Applications Conference*, Orlando, Florida, 1993.

[BCC94] N. Boulahia-Cuppens, F. Cuppens, A. Gabillon, and K. Yazdanian. Decomposition of Multilevel Objects in an Object-Oriented Database. In *European symposium on research in computer security*, Brighton, UK, 1994. Springer Verlag.

[BL75] D. Bell and L. LaPadula. Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, MTR-2997, MITRE, Bedford, Mass, 1975.

[CC95] L. Cholvy and F. Cuppens. Providing Consistent Views in a Polyinstantiated Database. In J. Biskup, M. Morgenstern, and C. Landwehr, editors, *Database Security, 8: Status and Prospects*. North-Holland, 1995. Results of the IFIP WG 11.3 Workshop on Database Security.

[CD89] F. Cuppens and R. Demolombe. How to recognize topics to provide cooperative answering. *Information Systems*, 14(2), 1989.

[DLS+87] D. Denning, T. Lunt, R. Shell, M. Heckman, and W. Shockley. A Multilevel Relational Data Model. In *IEEE Symposium on Security and Privacy*, Oakland, 1987.

[DLS+88] D. Denning, T. Lunt, R. Shell, W. Shockley, and M. Heckman. The SeaView Security Model. In *IEEE Symposium on Security and Privacy*, Oakland, 1988.

[GLQS92] T. Garvey, T. Lunt, X. Qian, and M. Stickel. Toward a Tool to Detect and Eliminate Inference Problems in the Design of Multilevel Databases. In *Proc. of the Sixth IFIP WG 11.3 Working Conference on Database Security*, Vancouver, 1992.

[JSS90] S. Jajodia, R. Sandhu, and E. Sibley. Update Semantics for Multilevel Relations. In *Sixth Annual Computer Security Applications Conference*, Tucson, Arizona, 1990.

[KM88] H. Katsuno and A. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52, 1988.

[Lun91] T. F. Lunt. Polyinstantiation: an inevitable part of a multilevel world. In *Proc. of the computer security foundations workshop*, Franconia, 1991.

[Rei83] R. Reiter. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*. Springer Verlag, 1983.

[SJ91] R. Sandhu and S. Jajodia. Honest Databases That Can Keep Secrets. In *Proceedings of the 14th National Computer Security Conference*, Washington, D.C., 1991.

[SJ92] R. Sandhu and S. Jajodia. Polyinstantiation for cover stories. In *European symposium on research in computer security*, Toulouse, France, 1992. Springer Verlag.

[Spa94] A. Spalka. Secure Logic Databases Allowed to Reveal Indefinite Information on Secrets. In *Database Security, 8: Status and Prospects*. North-Holland, 1994. Results of the IFIP WG 11.3 Workshop on Database Security.

[SW92] K. Smith and M. Winslett. Entity Modeling in the MLS Relational Model. In *Proceedings of the 18th International Conference on Very Large Data Bases*, Vancouver, Canada, 1992.

[Thu91] B. Thuraisingham. A Nonmonotonic Typed Multilevel Logic for Multilevel Secure Database / Knowledge-Based Management Systems. In *Proc. of the computer security foundations workshop*, Franconia, 1991.