データベース・システム 69－2
（1989. 1. 19）

多段階関係データベースにおけるデータ保護問題

チャットウィチェンチャイ ソムチャイ　　上林 彌彦

九州大学工学部

本稿では，関係データベースにおけるデータ保護の問題を一般的に扱うために，D.E. Denningらの多段階関係データベース（MDB）の概念を拡張する．次に，このモデルを用いて普遍関係データベースに対するデータ保護問題として重要な三つの問題を考え，それらの対策について検討する．第一の問題は，関数従属性や結合従属性を考慮した場合のデータ保護の問題で，単純な場合についてSuらの得た結果があるが，より一般的な場合についての検討を行った．第二の問題は，ある属性値の変更が許されなくても他の属性値を変更することにより等価的に禁止された変更を行なう問題であり，本稿ではじめて扱う．第三の問題は，アクセスクラスが属性の値を用いて定義されている場合，その属性の値を変えることによりアクセスクラスを変える問題である．この問題を従来のMDBモデルで扱うとデータベースアクセスに対する制限が非常に強くなるが，本稿のモデルによって制限を緩和することができる．

Preventing Inference and Unauthorized Modification of Protected Data in Multilevel Relational Databases

Somchai CHATVICHIENCHAI　Yahiko KAMBAYASHI

Department of Computer Science and Communication Engineering, Kyushu University

In this paper, we extend the concept of multilevel relational database (MDB) given by Denning's group in order to handle problems of protected data of relational databases. We consider 3 security problems which may arise when we use that concept to protect some data of a universal relation. The first is the problem of inferring some protected data by using the knowledge of functional and join dependencies that hold in the universal relation. Although this problem was first discussed by Su and Ozsoyoglu, their solution is not general enough. We give a more general solution of this problem. The second problem is that of unauthorized modification of some protected data. This problem arises when some users modify values of the attributes relevant to the protected data and the result of that modification is equivalent to modifying the protected data. This problem is first discussed in this paper. The last problem deals with modifying the access classes of some protected data in the case that access classes of the protected data are defined depending on the values of other attributes of the same tuple. This is a new problem which cannot be properly handled by the model of Denning's group since their solution will impose too much restriction of access of some data in the database. The solution proposed by our model gives less restriction than that of Denning's group.

1. Introduction

A multilevel relational database system (MDB) is a relational database system which stores data of different security classifications and provides these data to users with different access classes. Research about multilevel relational database systems was first initiated in 1982 by the group of Woods Hole Summer Study on multilevel Database Management Security. Since, then, the topic has attracted researchers' attention, and some important results have been derived [1], [2] and [3]. In Denning's paper[1], basic concepts for a multilevel-secure relational DB based on views is introduced. These concepts have since evolved into a multilevel relational model[2] that consists of multilevel relations with access classes assigned at the element level, a set of integrity constraints, and a method of decomposing all multilevel relations into single level relations. From the security managers' point of view, it is more convenient to assign access classes to data of a universal relation than to assign access classes to data of actual relations. The reason is, the universal relation is regarded as a conceptual relation representing the data of an entire database. However, there are problems that must be considered when access classes are assigned to data of the universal relation. The first problem is the inference problem due to integrity constraints (i.e., constraints that enforce the correctness of data in the database).

Recently, Su[3] consider FDs and a single multivalued dependency (MVD) as their basis for inference in a MDB. Their interesting work considers first how access classes of an attribute can be assigned in order to avoid inference via known FD mappings. For Example, if A functionally determines B, they assume that this mapping have already been known by users. Hence if B is protected with high access class, then they argue that access class of A must also defined at least as high as that of B, otherwise it can be used to permit to infer B. However this assumption limits their solution since most FD-mappings are not defined by an algorithm but rather than by the tuple instances (actually their projection onto the attributes of the FD). They also show how an MVD can pose a danger when access classes are assigned to tuple level. For Example, if protected tuples share the same MVD values with other tuples, the access classes of these other tuples may need to be defined as high as those of the protected tuples. However, this solution is not sufficient enough since the solution is based on access class of tuple level and MVD which is a special case of join dependency (JD). Hence, we give more general solution for inference of protected data due to usage of the knowledge of FDs and JDs in Sections 3 and 4.

The second problem is problem of unauthorized modification of protected data of MDB. This problem arise when security managers assign access classes to data by considering importance of individual attributes rather than that of attribute sets used to represent sensitive information. We discuss this problem in Section 4.

The last problem is problem of modifying access class of some protected data. Unauthorized users may be able to decrease access classes of protected data if the access classes are defined depending on the values of other data whose access classes are lower than that of the protected data. We discuss this problem in Section 5.

2. Basic Definitions

2.1 Access Classes

2.1.1 Access Classes of Denning's Model

An access class is a composite: (Classification levels, a set of categories). Fig. 1 shows classification levels and their order, and a few Example categories. The set of all possible access classes is structured as a lattice with partial ordering relation $\geqq$, called dominates[4]. An access class is said to be dominate another if and only if:

- its classification level is greater than or equal to the other, and
- its category set contains the other.

TOP-SECRET > SECRET > CONFIDENTIAL > UNCLASSIFIED

a) Classification Levels

Manufacturing, Personnel, Engineering, Accounting

b) Example Categories

Fig. 1: Classification Levels and Categories

For simplicity, we will fix category part but vary classification part of access class. Thus, access class will be specified simply as TOP-SECRET(TS), SECRET(S) etc.

In Denning's model, each datum D is assigned with an access class (let's C_D represent the access class of data D). Each user U is assigned with 2 access classes (O_U: operating access class, and W_U: the lowest operating access class of other users whom user U is permitted to release data to; where $O_U \geqq W_U$). Authorization control of Denning's model is defined as follows:

1) A user U is allowed to read data D if $O_U \geqq C_D$.
2) A user U is allowed to write data D if $C_D \geqq W_U$.

A user U is allowed to append only data D whose $R_D \geqq W_U$ such that he cannot release protected data to the other users who hold operating access classes lower than W_U. However, given data D, Denning's model lacks the ability to define the users who can only read data D and the users who

can both read and modify it. This problem will be explained in detail in Section 5.

2.1.2 Access Classes of Our Model

In our model, we improve Denning's model by assigning each data D with 2 access classes (readclass R_D - the lowest operating access class of the user who is permitted to read this data, and writeclass W_D - the lowest operating access class of the user who is permitted to update this data; where $W_D \geqq R_D$). Each user is assigned with the same access classes as defined in Denning's model. Authorization control of our model is defined as follows:-

1) A user U is allowed to read data D if $O_U \geqq R_D$.
2) A user U is allowed to append data D if $O_U \geqq W_D$ and $R_D \geqq W_U$.
3) A user U is allowed to delete data D if $O_U \geqq W_D$.
4) A user U is allowed to modify data D_1 to D_2 if $O_U \geqq W_{D_1}$, $O_U \geqq W_{D_2}$ and $R_{D_2} \geqq W_U$ (combination of 2) and 3)).

In Denning's model, a user U is allowed to write data D whose $C_D \geqq O_U$. This authorization is not suitable because user U cannot examine and adjust the data D after it has been written into the database. In our model, we solve this problem by allowing user U to append the data which he can modify.

2.2 A Multilevel Relation

In this model, a multilevel relation is enhanced from a standard relation by adding attributes used to represent access classes of data elements to the relation. Let $R(A_1, A_2, .., A_n)$ be a relation, R be a set of attribute in R. A multilevel relation for relation R is modeled by a schema $ML\_R(A_1, RC_{A_1}, WC_{A_1}, .., A_n, RC_{A_n}, WC_{A_n})$, where RC_{A_i} is classification attribute representing readclass of data attribute A_i, and WC_{A_i} is classification attribute representing writeclass of data attribute A_i. Users' view on ML_R is modeled as a schema $ML\_V(A_1, A_2, .., A_n)$. Fig. 2 illusttrates an instance of multilevel relation ML_R with three data attributes A, B and C. The data element whose readclass is not dominated by user's operating access class will appear as "NULL" in the user's view. However, if readclasses of all elements of a tuple are not dominated by user's operating access class, that tuple will not appear in the user's view. Fig. 3 shows data of ML_R seen by a secret user.

2.3. Classification Constraints

A classification constraint S is a rule that specifies value for classification attributes RC_{A_i} and WC_{A_i}. Formally, each rule S is a 4-tuple of the form: S = (ATTRS, EXP, TYP, CLS),

ML_R

A	RC_A	WC_A	B	RC_B	WC_B	C	RC_C	WC_C
a_1	C	C	b_1	S	S	c_1	S	S
a_2	S	S	b_1	TS	TS	c_1	S	S
a_3	S	S	b_2	S	S	c_2	TS	TS
a_4	TS	TS	b_3	TS	TS	c_3	TS	TS

Fig 2: Multilevel relation ML_R

ML_V

A	B	C
a_1	b_1	c_1
a_2	NULL	c_1
a_3	b_2	NULL

Fig 3: Multilevel view for secret users

where ATTRS is a list of one or more data attributes, EXP is an optional expression, TYP is type of access class (Readclass and/or Writeclass), and CLS is value of access class. The rule is interpreted as follows:

if EXP then TYP of ATTRS = CLS.

The expression EXP is a conjugation of one or more conditions to be satisfied by a collection of attributes in the database.

2.4 Policy of Assigning Access Class

Access classes are defined at the following three levels of granularity: attribute, tuple, and element level. In this paper, we investigate the security problems which may arise in case access classes are assigned at attribute, tuple, and element level.

3. Assigning Access Class at Attribute Level

3.1 Assigning readclass at Attribute Level

Attributes when are brought together through association provide more sensitive information than that of individual attribute. Sometimes there is a requirement to protect the information represented by association among attributes with readclass higher than that of each attribute in the association. Then, we divide readclasses assigned at attribute level into 2 types:

(a) readclasses defined for individual attributes
(b) readclasses defined for associations among attributes

Example 1: Consider relation EMP(NAME, POSITION, SALARY) with FD NAME → POSITION and POSITION → SALARY. If we want

to prevent a secret user from learning "the salaries of particular employees" from relation EMP while permit him to know "names and positions of employees in the company", then there are two methods for assigning readclasses to attributes of EMP.

Method 1: Assign at most SECRET to attributes NAME and POSITION, and assign TOP-SECRET to attribute SALARY.

Method 2: Assign at most SECRET to attributes NAME, POSITION and SALARY, and assign TOP-SECRET to association between NAME and SALARY. Then, secret users are prevented from obtaining values of attributes NAME and SALARY together under the same query request.

However, the first method gives different result from the second one. In the first method, secret users cannot know salary of any employees in the company. In the second method, secret users are permitted to know the list of salary values so that they can calculate salary statistics.

Readclass of an association among attributes can be defined by using the following classification constraints:

ASC_S_i = (*{ASC_ATR_LST}, , TYP, CLS),

where ASC_ATR_LST is a list of attributes in an association, and EXP is absent. Note that readclasses of associations among attributes are the abstract readclasses which are not store in a multilevel relation like readclasses of individual attributes.

In assigning readclass to associations among attributes, we must consider about inference of protected data represented by an association among attributes. The inference of the protected data can be done by reading authorized data and using the knowledge of FDs and JDs of the relation. Referring to Example 1, there would be no inference problem of protected data if there didn't exist FD POSITION → SALARY. Since secret user who know FDs of relation EMP can find out the data represented by association between NAME and SALARY by joining association between NAME and POSITION and association between POSITION and SALARY through POSITION. Therefore, it is necessary to assign TOP-SECRET to association between NAME and POSITION or association between POSITION and SALARY such that secret users cannot know association between NAME and SALARY by using the knowledge of functional and join dependencies which hold in relation EMP.

Let us now give formal definitions for an association among attributes and condition of inferring values of the protected association.

Let $R(A_1, A_2, .., A_n)$ be a universal relation scheme, r be any instance of R, and F be a set of FDs and JDs over R.

Definition 3.1.1: An association among attributes (ASC_ATR)

Let us represent an association among attributes to be protected by an attribute set ASC_ATR, and RC(ASC_ATR) be readclass of ASC_ATR. ASC_ATR is an attribute set which an access to $\Pi_X(r)$ where $X \supseteq$ ASC_ATR is prohibitted.

Note: $RC(ASC\_ATR) \geqq RC(ASC\_ATR')$ if and only if $ASC\_ATR \supset ASC\_ATR'$.

Let ASC_ATRS be a set of ASC_ATRs that have the same readclass.

Definition 3.1.2: JD-Compromise to ASC_ATRS under S

There exists JD-Compromise to ASC_ATRS under S if and only if there exists a $JD^*[\underline{R}_1, \underline{R}_2, .., \underline{R}_m\}$ implied by F such that $\Pi_X(\Pi_{\underline{R}_1}(r) * \Pi_{\underline{R}_2}(r) * \cdots * \Pi_{\underline{R}_m}(r)) = \Pi_X(r)$ where $X \in$ ASC_ATRS, $\underline{R}_i \in S$ ($1 \leqq i \leqq m$), $\underline{R} \supseteq \underline{R}_1 \cup \underline{R}_2 \cup \cdots \cup \underline{R}_m \supseteq X$, S be sets of attributes which users holding operating access class < RC(ASC_ATR) can access, and * represents a natural join operator.

We give the following algorithm to detect the existence of JD-Compromise to ASC_ATRS.

Algorithm for checking the existence of JD-Compromise to ASC_ATRS

Input: A universal relation scheme $R(A_1, .., A_n)$,
F which is a set of FDs and JDs over R,
$ASC\_ATRS = \{ASC\_ATR_1, .., ASC\_ATR_p\}$,
RC(ASC_ATRS) which is the readclass assigned to ASC_ATRS

Output: A decision whether there exist JD-Compromise to ASC_ATRS. If exist, return INFER_ASC which contains ASC_ATRS's members whose values can be known by using some JDs implied by F.

Method:

(1) Find S which is a set of attribute sets that users holding operating access class < RC(ASC_ATRS) can access.

$S = \{X \mid X \in 2^{\underline{R}} - \cup_{i \in (1,..,p)} SS(ASC\_ATR_i)\}$

$SS(ASC\_ATR_i) = \{X \mid X \subseteq \underline{R}, X \supseteq ASC\_ATR_i\}$

(2) Eliminate S's members which are proper subsets of the other member in S.

(3) Referring to the Chase Algorithm[5], construct a table with each column corresponding to each attribute of R and each row corresponding to each member in S. Apply the Chase Algorithm to compute inference of dependencies of F.

(4) After the Chase Algorithm finishes computation, if we discover that some row of the table contains $a_i\ a_j\ ..\ a_m$ each of which corresponds to each attribute of some ASC_ATR_i (of

ASC_ATRS), then there exists JD-Compromise to ASC_ATRS. If not, process stop.

(5) In this step, we will find ASC_ATR_i that $\Pi_{ASC\_ATR_i}(r)$ can be inferred by some JDs implied by F. Initialize ASC_INFER with {∅}. For each ASC_ATR_i in ASC_ATRS, if there is some row of the table contains a_i a_j .. a_m each of which corresponds to each attribute of ASC_ATR_i , then add ASC_ATR_i to ASC_INFER.

Now, we consider the associations among attributes that must be protected as well as each ASC_ATR_i of ASC_ATRS such that JD-Compromise to ASC_ATRS does not exist. Let's call those associations among attributes, a **JD-Compromise Inhibitor for ASC_ATRS**. JD-Compromise Inhibitor for ASC_ATRS is represented by a set of attribute sets ASC_IHB where $ASC\_IHB = \{ASC\_IH_1, .. , ASC\_IH_q\}$.

Definition 3.1.3: A JD-Compromise Inhibitor for ASC_IHB

ASC_IHB is a JD-Compromise Inhibitor for ASC_ATRS if there exists no JD-Compromise to ASC_ATRS under S where $S = 2^{\underline{R}} - \cup_{i \in (1, .. , p)} SS(ASC\text{-}ATR_i) - \cup_{i \in (1, .. , q)} SS(ASC\_IH_i)$.

In general, the may be more than one JD-Compromise Inhibitor for ASC_ATRS. Then, we give the following algorithm for finding a list of ASC_IHBs.

Algorithm for finding a list of ASC_IHBs

Input: A universal relation scheme $R(A_1\ A_2, .. , A_n)$,
F which is a set of FDs and JDs over R,
$ASC\_ATRS = \{ASC\_ATR_1, .. , ASC\_ATR_p\}$,
$S = 2^{\underline{R}} - \cup_{i \in (1, .. , p)} SS(ASC\text{-}ATR_i)$,
$ASC\_INFER = \{ASC\_INF_1, .. , ASC\_INF_s\}$
where ASC_INF_i is a member of ASC_ATRS, and its value is infered by some JDs implied by F.

Output: A list of ASC_IHBs

Method:

(1) Initialize JD_INFER = {∅}.

(2) For each $JD^*[\underline{R}_1, \underline{R}_2, .. , \underline{R}_m]$ implied by F, if (a) each $\underline{R}_k \in S$ $(1 \leqq k \leqq m)$ and (b) $\underline{R}_1 \cup \underline{R}_2 \cup .. \cup \underline{R}_m \supseteq$ some ASC_INF_j $(1 \leqq j \leqq s)$, then add $\{\underline{R}_1, \underline{R}_2, .. , \underline{R}_m\}$ as a member of JD_INFER.

Note: For simplicity, we assume that each $\underline{R}_k$ $(1 \leqq k \leqq m)$ is in Fifth Normal Form.

Suppose that JD_INFER obtained from this step consists of the following members: JD_INF_1, $JD\_INF_2, .. , JD\_INF_t$.

(3)Eliminate JD_INF_i from JD_INFER if there exist JD_INF_j such that $JD\_INF_i \supset JD\_INF_j$.

(4) An ASC_IHB is computed by taking a single member from each JD_INF_i (of JD_INFER) as a member for ASC_IHB. Therefore, there are several JD-Compromise Inhibitors for ASC_ATRS. However, it is better to find an ASC_IHB which has small number of members, and each member is a large attribute set in order to enable user access more attribute sets without inferring data of ASC_ATRS.

Example 2: Consider R = ABCD with F = {A → B, B → C, D → B}. Let ASC_ATRS = {BC, CD, ABD}. There exists JD-Compromise to ASC_ATRS due to the following JDs: {*[AB, AC], *[AB, AC, BD], *[AD, AC], *[AD, AC, BD], *[AD, AB]}. JD_INFER = {*[AB, AC], *[AD, AC], *[AD, AB]}. A list of ASC_IHBs for ASC_ATRS are {AB, AD}, {AB, AC}, {AC, AD}, and {AB, AC, AD}. However, it's better to select {AB, AD} or {AB, AC} or {AC, AD} as a JD-Compromise Inhibitor for ASC_ATRS.

3.2 Assigning Writeclass at Attribute Level

To prevent information from modifying by some users, we must consider writeclasses for attributes used to represent the information. However, information is represented by association among different attributes, then definition of writeclasses should be based on the association among attributes rather than based on individual attributes. Defining writeclass based on association among attributes provides the precise scope of relevant attributes which must be considered better than defining writeclass based on individual attributes.

Example 3: Consider relation DEPOSIT(ACC#, NAME, ACC_DATE, BAL). Association between ACC# and BAL provide information "balances of customer accounts". To prevent secret users from modifying balances of customer accounts, we usually think that defining writeclass of attribute BAL to be TOP-SECRET is enough. In fact, it is not secure at all. Let both readclass and writeclass of attribute ACC#, NAME, ACC_DATE be S, readclass and writeclass of attribute BAL be S and TS respectively. Suppose that t_1(01201 KATO 85.12.05 1,500,000) and t_2(02514 TANAKA 87.09.15 8,000,000) are tuples of relation DEPOSIT. Although writeclass of BAL is TOP-SECRET, secret user can modify balance of account 01201 to be 8,000,000 by swapping account number of t_1 with that of t_2. In order to make it difficult to be detected, he may also swap name and account date of t_1 with those of t_2. To prevent secret users from modifying balances of customer accounts effectively, writeclass of association between ACC# and BAL must be defined to be TOP-SECRET. However, in case of writeclass, assigning TOP-SECRET to association between ACC# and BAL is equivalent to assigning TOP-SECRET to attribute ACC# and BAL. Then we can apply standard classification constraints for defining writeclass of association among attributes. Writeclass of an association

among attributes is defined by the following classification constraints:

S_i = ({ASC_ATR_LSR}, , {W}, CLS),

where ASC_ATR_LST is a list of attributes in an association, EXP is absent.

In general, there may be some attributes participating in more than one associations, and some of those associations are assigned with different writeclasses. Thus, it is necessary to select the effective writeclass for the attribute (which participates in those associations) from the writeclasses assigned to those associations. In this paper, the effective writeclass of an attribute is defined to be equal to the least upper bound of writeclasses assigned to the associations in which the attribute is participating. After effective writeclass of each attribute in the relation is computed, the effective writeclasses of associations among attributes will be considered. Formal definitions for the effective writeclasses of an association is as follows:

Let's represent an association among attributes by an attribute set ASC_ATR where ASC_ATR = $\{A_1, A_2, .. , A_k\}$. Let EF_WC(A_i) be the effective writeclass of attribute A_i, EF_WC(ASC_ATR) be the effective writeclass of ASC_ATR.

EF_WC(ASC_ATR) = $\otimes${EF_WC(A_1), EF_WC(A_2), .. , EF_WC(A_k)}

where $\otimes$ denotes the greatest lower bound operator.

Example 4: Referring relation DEPOSIT of Example 3, let consider the effective writeclass of each attribute and the effective writeclasses of associations among attributes from the following classification constraints:

S_1 = ({ACC#, NAME, DATE}, , {W}, S)

S_2 = ({ACC#, NAME, BAL}, , {W}, TS)

The effective writeclasses of ACC#, NAME, DATE and BAL are TS, TS, S and TS respectively.

EF_WC({ACC#, NAME, DATE}) = $\otimes${TS, S} = S

EF_WC({ACC#, NAME, BAL}) = TS

4. Assigning Access class at Tuple Level

In this case, all data elements associated with a tuple have the same access class which is determined by the values of some data elements of the tuple. However, readclass of a tuple of a relation must also be defined with regard to JDs that hold in the relation. Otherwise, those users may be able to infer some data of protected tuples by reading tuples authorized to them and utilizing the knowledge of JDs of the relation.

Example 5: Suppose relation R(ABC) satisfies the JD*[AB,BC,AC] and an instance of multilevel relation for R contains the four tuples shown in Fig. 4. ML_r shown in Fig. 5 is a secret user's view upon ML_R. Although secret user cannot read tuple (a_1 b_1 c_1), however he can know the existence of this tuple by joining the projection of AB, BC and AC over ML_V. Note that tuple (a_1 b_1 c_1) shares values of AB, BC, and AC with the first tuple, the second tuple, and the third tuple respectively. In this case, we say there is inference of protected tuples by JD*[AB,BC,AC] which applies to R.

Now, we give formal definition for inference of some protected tuple(s) by JDs that hold in R and a necessary and sufficient condition to prevent the inference.

ML_R

A RC_A WC_A	B RC_B WC_B	C RC_C WC_C
a_1 C S	b_1 C S	c_2 C S
a_2 S S	b_1 S S	c_1 S S
a_1 S S	b_2 S S	c_1 S S
a_1 TS TS	b_1 TS TS	c_1 TS TS

Fig 4: Multilevel relation ML_R

ML_V

A	B	C
a_1	b_1	c_2
a_2	b_1	c_1
a_1	b_2	c_1

Fig 5: Multilevel view for secret users

Definition 4.1: Inference of some data of protected tuple by JD*[$\underline{R}_1$, $\underline{R}_2$, .. ,$\underline{R}_m$]

Let R(A_1, A_2, .. , A_n) be a universal relation, F be a set of FDs and JDs over R, ML_R(A_1, RC_{A_1}, WC_{A_1}, .. , A_n, RC_{A_n}, WC_{A_n}) be a multilevel relation for R, and ML_r be any instance of ML_R.

Let T_{low} = { t | t $\in$ ML_r, RC(t) $\leqq$ O_U},

T_{high} = { t | t $\in$ ML_r, RC(t) > O_U},

where RC(t) is readclass assigned to all elements of tuple t,

O_U is an operating access class of a user.

There exists inference of some data of protected tuples of ML_r by JD*[$\underline{R}_1$, $\underline{R}_2$, .. ,$\underline{R}_m$] implied by F where $\underline{R}_1 \cup \underline{R}_2 \cup .. \cup \underline{R}_m \subseteq \underline{R}$ if and only if

$\Pi_{\underline{R}_1}(T_{low}) * \Pi_{\underline{R}_2}(T_{low}) * \cdots * \Pi_{\underline{R}_m}(T_{low}) = \Pi_{\underline{R}_1 \cup \underline{R}_2 \cup .. \cup \underline{R}_m}(T_{low} \cup T')$

where $T' \subseteq T_{high}$

Lemma 4.1: A necessary and sufficient condition for preventing inference of data of protected tuples due to JD*[$\underline{R}_1$, $\underline{R}_2$, .. ,$\underline{R}_m$]

There exists no inference of some data of protected tuples of ML_r if and only if

$(\forall i)(\forall O_U)(\exists t_i \in T_{low})(\neg\exists t_k \in T_{high})$:

$t_k[\underline{R_i}] = t_i[\underline{R_i}]$ where $1 \leqq i \leqq m, k \neq i$.

Proof: Referring to definition of JDs in a manner similar to the definition of MVD[5], we consider $JD^*[\underline{R_1}, \underline{R_2}, .., \underline{R_m}]$ that applies to r as follows: If r contains tuple $t_1, t_2, .., t_m$ such that $t_i[\underline{R_i} \cap \underline{R_j}] = t_j[\underline{R_i} \cap \underline{R_j}]$ for all i and j, then r must contain a tuple t_k such that $t_k[\underline{R_i}] = t_i[\underline{R_i}]$, $1 \leqq i \leqq m$, $k \neq i$. Although a user is not permitted to access t_k, but he can obtain it if he can access all t_i's($1 \leqq i \leqq m$). Then, there is inference of t_k by $JD^*[\underline{R_1}, \underline{R_2}, .., \underline{R_m}]$ if and only if $t_k \in T'$ and all t_i ($1 \leqq i \leqq m$) $\in T_{low}$. In order that there exists no inference of t_k by $JD^*[\underline{R_1}, \underline{R_2}, .., \underline{R_m}]$, t_k must not be in T_{high} if all t_i's($1 \leqq i \leqq m$) are in T_{low}.

5. Assigning Access Classes at Element Level

In this case, a data element of a tuple may be assigned with readclass (and/or writeclass) which may be different from those of other data elements of the tuple. Readclass and writeclass of a data element are defined depending on its value or on the values of other data elements of the same tuple.

5.1 Inference of some protected data elements by JDs

As we have explained in Section 4, there may be inference of protected data by using the knowledge of JDs, if readclasses of tuples are defined with regardless to the JDs that hold in the relation. However this problem becomes complex in case access class are assigned at data element level. Let's now describe the definition of inference of protected data elements by using the JDs which hold in a universal relation.

Definition 5.1: The least upper bound of access classes of the elements of data attribute set X in tuple t_p of relation ML_r

Let ML_R, ML_r be defined as Definition 4.1, and let $X = \{A_{x_1}, A_{x_2}, .., A_{x_m}\}$ be a subset of attributes in R, and $t_p[RCU_X]$ be the least upper bound of access classes of the elements of data attribute set X in tuple t_p of relation ML_r.

$t_p[RCU_X] = \oplus \{t_p[RC_{A_{x1}}], t_p[RC_{A_{x2}}], .., t_p[RC_{A_{xm}}]\}$

where $\oplus$ denote the least upper bound operator, and $x_i = 1, .., n$

Definition 5.2: Inference of some protected data elements by $JD^*[\underline{R_1}, \underline{R_2}, .., \underline{R_m}]$

Let R, F, ML_R, and ML_r be defined as Definition 4.1. Consider $JD^*[\underline{R_1}, \underline{R_2}, .., \underline{R_m}]$ implied by F where $\underline{R'} = \underline{R_1} \cup \underline{R_2} \cup .. \cup \underline{R_m} \subseteq \underline{R}$.

Let $T[\underline{R_i}] = \Pi_{\underline{R_i}}(ML\_r)$,

$T_{low}[\underline{R_i}] = \{t \mid t \in T[\underline{R_i}], RCU_{\underline{R_i}} \leqq O_U\}$,

$T_{high}[\underline{R_i}] = \{t \mid t \in T[\underline{R_i}], RCU_{\underline{R_i}} > O_U\}$,

$T[\underline{R'}] = \Pi_{\underline{R'}}(ML\_r)$,

$T_{low}[\underline{R'}] = \{t \mid t \in T[\underline{R'}], RCU_{\underline{R'}} \leqq O_U\}$,

$T_{high}[\underline{R'}] = \{t \mid t \in ,T[\underline{R'}]\, RCU_{\underline{R'}} > O_U\}$

There exists inference of some protected data elements of ML_r by $JD^*[\underline{R_1}, \underline{R_2}, .., \underline{R_m}]$ implied by F where $\underline{R'} = \underline{R_1} \cup \underline{R_2} \cup .. \cup \underline{R_m} \subseteq \underline{R}$ if and only if

$T_{low}[\underline{R_1}] * T_{low}[\underline{R_2}] * \cdots * T_{low}[\underline{R_m}] = T_{low}[\underline{R'}] \cup T'$ where $T' \subseteq T_{high}[\underline{R'}]$

Lemma 5.1: A necessary and sufficient condition for preventing inference of protected data elements due to $JD^*[\underline{R_1}, \underline{R_2}, .., \underline{R_m}]$

There exists no inference of some data of protected data elements of ML_r if and only if

$(\forall i)(\forall O_U)(\exists t_i[\underline{R_i}] \in T_{low}[\underline{R_i}])$

$(\neg\exists t_k[\underline{R_i}] \in T_{high}[\underline{R_i}])$: $t_k[\underline{R_i}] = t_i[\underline{R_i}]$ where $1 \leqq i \leqq m, k \neq i$.

Proof: Similar to the proof of Lemma 4.1.

Besides inference of protected data elements by some JDs, there may also be inference of some protected data elements if readclasses of data elements are defined with regardless to FDs the hold in a relation. We call this type of inference, "inference of some protected data elements by FDs".

Definition 5.3: Inference of some protected data elements by FD $X \rightarrow A_i$

Let F be a set of FDs on a relation R. Consider a FD $X \rightarrow A_i$ implied by F where $A_i \notin X$. There exists inference of some protected data elements by FD $X \rightarrow A_i$ implied by F if and only if

$(\exists t_p \in ML\_r)(\exists t_q \in ML\_r)(\exists O_U)$ s.t.

$t_p[X] = t_q[X]$ and

$t_p[RCU_X] \leqq O_U$ and $t_p[RC_{A_i}] \leqq O_U$ and

$t_q[RCU_X] \leqq O_U$ and $t_q[RC_{A_i}] > O_U$, where

RC_{A_i}, RC_{A_j} are attributes denoting readclass of A_i, and A_j repectively,

O_U is operating access class of user U, and

$i = 1, 2, .., n$

Lemma 5.2: A Sufficient condition for preventing inference of protected data elements due to FD $X \rightarrow A_i$

There exists no inference to protected data by FD $X \rightarrow A_i$ implied by F where $A_i \notin X$, if

$(\forall t_q \in ML\_r)(\forall O_U)(\exists A_k \in X)$: $t_q[R_{A_k}] \geqq t_q[R_{A_i}]$

Proof: Since $A_k \in X$ and $RC_{A_k} \geqq RC_{A_i}$, then $t_q[RCU_X] \geqq t_q[RC_{A_k}] \geqq t_q[RC_{A_i}]$. This constraint does not satisfy the condition of Definition 5.3 which defines the existence of a tuple t_q having $t_q[RC_{A_i}] > O_U \geqq t_q[RCU_X]$. Therefore, there exists no inference of protecetd data element by FD $X \rightarrow A_i$ implied by F.

5.2. Preventing Modification of Access Classes of Some Protected Data

In a multilevel relation, access classes of a data element may be defined depending on its value or on the values of other data elements of the same

tuple. However, there may be problems of disclosure and modification of protected data if access classes of the protected data are defined depending on the values of data whose writeclasses are lower than those of the protected data. Then, some users may be able to disclose or modify the protected data by modifying values of the data used to determine access classes of the protected data.

Example 6: Let's consider readclasses and writeclasses defined for attributes of relation EMP(NAME, POSITION, SALARY) through the following classification constraints:

S_1 = ({NAME, POSITION, SALARY}, , {R, W}, S).

S_2 = ({SALARY}, POSITION = "Manager", {R, W}, TS).

According to classification constraint S_2, readclasses and writeclasses of the salaries of managers are defined to be TOP-SECRET. However, secret users can read and modify the salary of a manager by modifying position of the manager to other position. To solve this problem, the writeclass of position of a manager must be assigned to be TOP-SECRET.

This problem concerns with the reason why our model improves the access class of Denning's model by the readclass and writeclass. In Denning's model, a data element is associated with a single access class. If we solve the above problem with the solution based on the access class of Denning's model, then we must assign access class of salary of the manager to TOP-SECRET. However, this solution incurs an undesirable effect. That is, secret users cannot read positions of the employee who is a manager. Compare with the solution of our model which is based on readclass and writeclass, secret users can still read position of the employee who is a manager, but they cannot modify it. Therefore, our new type of access classes (readclass and writeclass) is better than the access class of Denning's model.

To prevent modification of access class of protected data as shown in the above example, writeclasses of all the data elements whose values are used to determine access class of an protected element must be defined to be at least as high as that of the protected element. However, an data element may be used to determine access classes of several protected elements. Futhermore, there may exist classification constraints defining different writeclasses for that data element. So, we need to define rules that determine which writeclass to be assigned to a data element. The rules are defined as follows:-

Let ML_R, ML_r be defined as Definition 4.1,

t_s be a tuple of a multilevel relation,

$t_s[A_i]$ be value of attribute A_i on tuple t_s,

$t_s[RC_{A_i}]$, $t_s[WC_{A_i}]$ be readclass and writeclass of $t_s[A_i]$ respectively,

$S_{all} = \{S_1, S_2, .. , S_m\}$ be a set of all classification constraints, and

LOGIC(EXP) be logical value of EXP. If EXP is absent or condition of EXP is satisfied, LOGIC(EXP) = TRUE, otherwise LOGIC(EXP) = FALSE.

$t_s[WC_{A_i}] = \oplus \{L_1, L_2, .. , L_q, t_s[RC_{A_i}]^{(1)}\}$, where L_j is an access class defined by constraint S_k where

(2) $S_k \in S_{all}$ s.t. $A_i \in ATTRS_k$, writeclass $\in TYP_k$, and LOGIC(EXP_k) = TRUE;

or (3) $S_k \in S_{all}$ s.t. there exists A_i in EXP_k, and LOGIC(EXP_k) = TRUE;

(where j = 1,2, .. ,q; k = 1,2, .. ,m).

Writeclass of a data element of attribute A_i is equal to the least upper bound of (1) readclass of that data element ($t_s[RC_{A_i}]$), (2) access classes of the constraints which effectively define writeclasses for that data element, and (3) access classes of the constraints which effectively define readclasses or writeclasses of other data elements depending partially or solely on the value of that data element.

6. Conclusion and Future Works

In this paper, we have shown that FDs and JDs play important roles in defining readclasses for the data stored in the universal relation. To prevent unauthorized modification of protected data effectively, definition of writeclass for data must be based on associations among attributes. In case access classes of protected data are defined depending on values of some elements, writeclasses of those elements must be defined to be at least as high as access classes of those protected data.

The concept of this paper will be used to develope an algorithm for checking whether the classification constraints given by a security manager define suitable access classes for data of the universal relation in such a way that there is no problem of inference of protected data.

References

[1] D.E. Denning et.al., "Views for multilevel database security", IEEE Trans on S/W Eng., SE-13(2): 129-140, Feb 1987

[2] D.E. Denning et.al., "A Multilevel Relational Data Model", Proc. of 1987 Symp. on Security and Privacy, IEEE Computer Society, 1987, pp. 220-233

[3] T. Su, G. Ozsoyoglu., "Data Dependencies and Inference Control in Multilevel Relation Database Systems", Proc. of the 1987 Symp. on Security and Privacy, IEEE Computer Society, 1987, pp. 202-211

[4] D.E. Denning, Cryptography and Data Security, Addision-Wesley, Reading, Mass., 1982

[5] D. Maier, The Theory of Relational Databases, Computer Science Press, 1983