A RELATIONAL DATABASE VIEW UPDATE TRANSLATION MECHANISM

Yoshifumi Masunaga

University of Library and Information Science
Yatabe-machi, Tsukuba-gun, Ibaraki-ken 305, Japan

ABSTRACT: A semantic approach to design a view update translator for relational database systems is presented in this paper. Our translator consists of a translator body and four different types of semantic ambiguity solvers. Since a view is defined as a tree with the view on the root and its base relations on the leaves, an update issued against the root can be translated into updates against the lower levels by applying a total of ten local translation rules and a deletion and an insertion modification rule recursively. The modification rules make it possible to update base relations through natural join views. Three of the ten local translation rules require three different types of semantic ambiguity solvers, and the two modification rules together require another solver. The translation capability depends on the solvers available to the translator body and the problem solving capability they offer. From the nature of such ambiguities, the solvers may involve the end-users in resolving the ambiguities.

1. INTRODUCTION

A view is a virtual relation derived from base (i.e., stored) relations using a set of view defining operations such as projection, join, and others (1). There are two major reasons why it is desirable to support views in a database system: The first is for user convenience, in the sense that the user can define his own database view to which he can issue queries and updates (2,3). Another is to provide an authorization mechanism (4,5).

The main purpose of this paper is to present and motivate a mechanism for propagating updates against views to their base relations. Unfortunately, view update capabilities are still not well supported in existing systems. We still lack a theoretical basis for designing an update translator of a database system. Also, existing systems allow only a subset of all theoretically possible views and view updates. For example, a deletion from a union view is always possible, but union views are not supported in SQL/DS (7). The purpose of this paper is to find a solution to the theoretical problem.

In general, a view is virtual and an update against a view is only possible if there exists a certain set of update(s) to the base relations. There are at least two major problems which must be solved. First is to determine which classes of view updates are possible and which are not. Second is to implement view updates. In 1974, Codd first reported the view update problem (1). Next year Chamberlin et.al. (2) and Stonebraker (3) proposed first-cut solutions to the problem. Then Paolini and Pelagatti (8) and Dayal and Bernstein (9) tried to formulate the problem. Furtado et.al. (10) and Osman (11) deduced a set of view update translation rules. It is now recognized that the problem is closely related to the database semantics. That is, it has been recognized by several authors that extra, semantic information should be supplied in order to resolve the anomalies which may arise in updating views. Dayal and Bernstein (9,19) and Carlson and Arora (13) adopted functional dependencies; Bancilhon and Spyratos (14,15,16) introduced the concept of complementary views; and Keller (18) used a structural data model in order to provide such

Proceedings of the Tenth International Conference on Very Large Data Bases.

Singapore, August, 1984

semantic information. However, these approaches are still insufficient because the view update translation must ultimately be guided by the user's intention when he issued the view update.

Our approach is based on analyzing the meaning of views. That is, we take into consideration the meaning of a view which is definable as a time-varying first order predicate calculus expression. Since the definition of a view is represented as a tree, where the view is on the root and its base relations are on the leaves, our view update mechanism translates an update to a view into the lower level views recursively by using translation rules. A total of ten local translation rules and a deletion and an insertion modification rule will be introduced. Each of the ten local translation rules has a semantical basis justified by the meaning of the view. Three of the ten rules require semantic ambiguity solvers when they are applied. Sometimes, these solvers may require interaction with the user. The two modification rules together also require another semantic ambiguity solver. This solver is necessary to resolve the semantic ambiguity which may arise when one updates natural join views. Therefore, our view update translator consists of total five components; a translator body and four semantic ambiguity solvers of different types. The translator body has an interface to each of the four solvers. The translation capability changes depending on what solvers are available to the body and how powerful they are. From the nature of such ambiguities, the solvers may involve the users in resolving the ambiguities.

The rest of this paper is organized as follows: A formal definition of views is given in section 2. In section 3, the meaning of views is formally defind. In section 4, view updatability criteria are re-examined. In section 5, a total of ten local view update translation rules are developed and motivated and a deletion and an insertion modification rule are presented. Four types of semantic ambiguity problems will be explained. Then, our translation mechanism will be made clear.

2. VIEWS

Let A1, A2, ..., An be attributes. Let dom be a function which associates each attribute Ai with its domain, dom(Ai). A relation R(A1,A2,...,An) (with respect to this domain function) is a finite subset of the direct product dom(A1)xdom(A2)x...xdom(An). We sometimes use R instead of R(A1,A2,...,An) and abbreviate the direct product to dom(R). By att(R) we denote the unordered set {A1, A2, ..., An}. We define views in terms of the relational algebra (17). There are four traditional set operations; direct product, union, intersection, and difference. Also there are four operations that are specific to the relational algebra, namely, projection, θ-restriction, division, and θ-join, where θ represents a comparison operator (=, >, <, ¬=, >=, <=). Of course, those eight operations are redundant and we select direct product, union, difference, projection, and θ-restriction as a generating set of the relational algebra. Then views can be defined as follows:

Definition:

(1) A base relation is a view.

(2) Let V be a view and X be a subset of att(V). Then the projection of V on X, denoted by V[X], is a view. Next, let X and Y be subsets of att(V) which are θ-compatible, where θ is a comparison operator. Then the θ-restriction of V on X and Y, denoted by V[XθY]*, is a view.

(3) Let V and W be views. Then the direct product of V and W, denoted by VxW, is a view. If V and W are union-compatible, then the union of V and W, denoted by VuW, and the difference of V and W, denoted by V-W, are views.

(4) A relation is a view if and only if it is derived by using the above three rules.

Notice that the views which could be defined by introducing the so-called virtual columns (7) are excluded in this definition because it is almost obvious that those views are impossible to update.

Now it is easy to see that we can construct a tree from the defining expression of a view, where the root and the leaves represent the view and the base relations which are used to define the view, respectively. We call a node which is neither the root nor a leaf an intermediate node. For example, suppose a parts dealer has a supplier-part-customer database which has two base relations, SP(supplier,part) and PC(part,customer). Then the view derived from a natural join of SP and PC, and denoted by SPC(supplier,part,customer), is defined as follows in our framework:

SPC=((SPxPC)[SP.part=PC.part])[supplier, SP.part,customer].

---

* Suppose X={Ai1,Ai2,...,Aip} and Y={Aj1,Aj2,...,Aip} are θ-compatible subsets of att(V), where Aik and Ajk are single attributes for every k=1,2,...,p. Then V[XθY] consists of all tuples v of V such that the predicate v[Aik] θv[Ajk] is true for every k.

Figure 1 (a) shows the view defining tree of SPC and 1 (b) shows instances of SP, PC, SPC and intermediate views named V0.1 and V0.1.1.

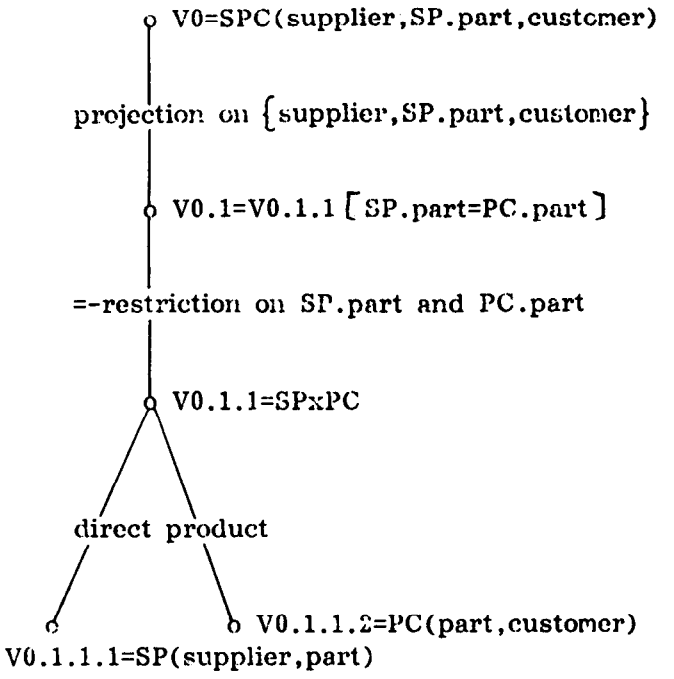

Figure 1 (a). The view defining tree of SPC.

SP:

supplier	part
s1	p1
s2	p2
s3	p1
s3	p2
s3	p3

PC:

part	customer
p1	c1
p1	c2
p2	c2

V0.1.1:

supplier	SP.part	PC.part	customer
s1	p1	p1	c1
s1	p1	p1	c2
s1	p1	p2	c2
s2	p2	p1	c1
s2	p2	p1	c2
s2	p2	p2	c2
s3	p1	p1	c1
s3	p1	p1	c2
s3	p1	p2	c2
s3	p2	p1	c1
s3	p2	p1	c2
s3	p2	p2	c2
s3	p3	p1	c1
s3	p3	p1	c2
s3	p3	p2	c2

V0.1:

supplier	SP.part	PC.part	customer
s1	p1	p1	c1
s1	p1	p1	c2
s2	p2	p2	c2
s3	p1	p1	c1
s3	p1	p1	c2
s3	p2	p2	c2

SPC:

supplier	SP.part	customer
s1	p1	c1
s1	p1	c2
s2	p2	c2
s3	p1	c1
s3	p1	c2
s3	p2	p2

Figure 1 (b). Instances of SP, PC, V0.1.1, V0.1, and SPC.

3. MEANING OF VIEWS

A formal definition of the meaning of views will be given as a time-varying first order predicate calculus expression. The reader may note that such an expression gives us a semantic basis from which we will deduce the local translation rules of view updates in section 5.

We define the meaning of a view as follows: For example, suppose V is a difference view, i.e. V=U-W, where U and W are views or base relations which are union-compatible. Then the meaning of V, denoted by M-of-V, may be characterized in terms of the meanings of U and W in such a way that

$(\forall t \in dom(V))(M\text{-of-}V(t)=(M\text{-of-}U(t)\ AND\ \neg M\text{-of-}W(t)))$.

The rationale for this definition is the observation that a tuple t of dom(V) satisfies M-of-V if and only if it satisfies M-of-U and not M-of-W. That is, t is a tuple of V if and only if it is a tuple of U and it is not a tuple of W.

The meanings of four other views can be defined in the same manner. Table 1 shows how the meanings of the five basic views may be defined in terms of the meaning of direct descendant views or base relations. Since an instance of a relation usually changes from time to time, a time-varying predicate calculus is adopted. The meaning of a base relation R, denoted by M-of-R, is that for any tuple t of dom(R), M-of-R(t) is true if and only if t belongs to R at this time. The meaning of any view is obtained by applying (M-1), ..., (M-5)

to the meanings of its base relations recursively.

4. VIEW UPDATABILITY CRITERIA

A view update is realizable if there exists a set of update(s) against the base relations which causes the intended update. But what does this really mean? One answer was given by Dayal and Bernstein (9) from a syntactic point of view. They impose essentially the following three criteria:

(Cri-1) No overupdating or underupdating of a view. This means that any tuple of the view should not be deleted or inserted or replaced unless it is specified to be done so by the view updator.

(Cri-2) No extraneous updates against the base relations. This means that any unnecessary base relation updates are not allowed to realize the desired view update.

(Cri-3) Unique view update translation.

We also adopt (Cri-1) and (Cri-2). However, we relax criterion (Cri-3). The reason is that although their interpretation is pertinent to formulating the view update translation problem in a syntactic way, it can only deal with a small subset of the semantic problems which arise in updating views. While Dayal and Bernstein interpreted Cri-3 syntactically, we interpret it semantically. More precisely, they say that the view update translation is unique if no more than one possible translation can be found when the syntactic information deduced from the definition of database schema is used for the translation. We say, however, that the translation is also unique if there exists a way to resolve the translation ambiguity by using certain semantic information. For example, suppose a user issues a deletion of a set of four tuples {(s1,p1,c1), (s1,p1,c2), (s3,p1,c1), (s3,p1,c2)} from view SPC. Then we can observe that there are three alternative base relation updates which can satisfy the view deletion as desired:

(Alt-1) Delete {(s1,p1), (s3,p1)} from base relation SP.

(Alt-2) Delete {(p1,c1), (p1,c2)} from base relation PC.

(Alt-3) Delete {(s1,p1), (s3,p1)} and {(p1,c1), (p1,c2)} from base relations SP and PC, respectively.

Notice that each alternative satisfies criteria (Cri-1) and (Cri-2), but arbitrary choice is unacceptable because it may result in a semantically inconsistent database state.

There are two ways to handle this situation: First, we might conclude that the view deletion is impossible because of this ambiguity. Second, we might allow that the deletion depends on whether this ambiguity is removable or not. The former is the approach adopted by Dayal and Bernstein, and the latter is one that we are adopting in this paper. In general, the latter alternative will give a more powerful view support capability than the former one. But we need some additional functions to perform semantic information processing. The precise mechanism for realizing our view support will be given in the next section. Suffice it to say for now that our approach will require four types of problem solvers which can resolve such ambiguity problems. In some cases, the ambiguity solvers may ask the user of his intention; (Alt-1), (Alt-2), or (Alt-3). Normally, however, it is

Type of View	Meaning of View
V=UxW Direct Product View	$(\forall t \in dom(V))(M\text{-of-}V(t)=(M\text{-of-}U(t[att(U)]) \text{ AND } M\text{-of-}W(t[att(W)]))).$ ...(M-1)
V=UuW Union View	$(\forall t \in dom(V))(M\text{-of-}V(t)=(M\text{-of-}U(t) \text{ OR } M\text{-of-}W(t))).$ ...(M-2)
V=U-W Difference View	$(\forall t \in dom(V))(M\text{-of-}V(t)=(M\text{-of-}U(t) \text{ AND } \neg M\text{-of-}W(t))).$ ...(M-3)
V=U[X] Projection View	$(\forall t \in dom(V))(M\text{-of-}V(t)=(\exists u \in dom(U))(u[X]=t \text{ AND } M\text{-of-}U(u)=true)).$ ...(M-4)
V=U[XθY] θ-restriction View	$(\forall t \in dom(V))(M\text{-of-}V(t)=(M\text{-of-}U(t) \text{ AND } t[X]\theta t[Y])).$ ...(M-5)

Table 1. Meaning Expressions of Five Basic Views

expected that the ambiguity solvers can use some semantic integrity constraints or specific knowledge given to them to resolve the problems.

5. VIEW UPDATE TRANSLATION MECHANISM

5.1 FUNDAMENTAL TRANSLATION SCHEME

As mentioned in Section 2, the definition of a view may be represented as a tree. We say that the root of the view tree is on level zero. The direct descendants are on level one, and so on. In the example in Figure 1 (a), view SPC is on level 0, the intermediate views V0.1 and V0.1.1 are on levels 1 and 2, respectively, and the base relations SP and PC are on level 3.

Suppose an update M is issued against the view V. Then our translation mechanism translates M into update(s) to its direct descendant(s). Of course, the three updatability criteria presented in section 4 must be satisfied in this translation. We call this type of translation a local translation and the corresponding translation rule a local translation rule. A total of ten local translation rules will be presented in detail in the next section. If the local translation is successful on one level of the view tree, local translation is performed at the next lower level recursively. If translation is successful on every level of the tree, the resulting updates to the base relations are the desired translation result. If local translation becomes impossible on any level, it will be concluded that view update M is unacceptable. Although success of local translation on each level of the view tree implies the success of the view update, there exists one (and only one) exception. That is, there exists one case in which an update to a certain type of intermediate view which is impossible will be allowed to be modified so that the modified update becomes translatable to a lower level. This rule is called the update modification rule and it will be discussed in section 5.3.

5.2 LOCAL TRANSLATION RULES

We will consider all five types of views: direct product view, union view, difference view, projection view, and θ-restriction view. An update against a view may be a deletion, an insertion, or a replace. However, replace can be realized as a deletion and insertion pair. Therefore, the local translation rules for only the deletion and insertion will be presented for each type of view. These rules will be justified by the meanings of views given in Table 1.

5.2.1 LOCAL TRANSLATION RULES FOR DELETION

Rule D-1 (Deletion against a Direct Product View): "Suppose deletion D is issued against a direct product view V=UxW. Then D can be translated into deletions D1 against U and D2 against W, respectively, where D1=(U-(V-D)[dom(U)]) and D2=(W-(V-D)[dom(W)]) if and only if the cross reference condition (see below) holds for the difference V-D."

Justification: Suppose deletion D is issued against a direct product view V=UxW, where D represents a set of tuples that the user wants to delete from V. We assume without loss of generality that D is a subset of V because V-D=V-(V∩D). Since the expected result relation V-D must again be a direct product (if the deletion D from V is realizable), the following condition called the cross reference condition must hold.

$$(\forall t,t' \in dom(V)(t,t' \in V-D \Rightarrow t[dom(U)] \| t'[dom(W)] \in V-D),$$

where ⇒ stands for logical implication and ‖ tuple concatenation operation (17). This means that whenever two tuples t and t' exist in V-D, the concatenation of the dom(U) part of t and the dom(W) part of t' must belong to V-D if it is a direct product. From expression (M-1) in Table 1, let us translate D into deletions D1 and D2 against U and W, respectively as follows:

D1=(U-(V-D)[dom(U)]).
D2=(W-(V-D)[dom(W)]).

Then it is proved that V-D=(U-D1)x(W-D2). That is, the cross reference condition is a necessary and sufficient condition for deletion D against V (=UxW) to be realizable under the above defined translation. □

Rule D-2 (Deletion against a Union View): "Suppose deletion D is issued against a union view V=UuW, where U and W are union-compatible. Then D is always translatable into deletions D1 and D2 against U and W, respectively, where D1=D2=D. "

Justification: Suppose deletion D is issued against a union view V=UuW, where U and W are union-compatible. Remember that the meaning of V is defined by expression (M-2) in Table 1 as

$$(\forall t \in dom(v)(M\text{-of-}V(t)=(M\text{-of-}U(t) \text{ OR } M\text{-of-}W(t))).$$

Since each tuple of D has lost the meaning of V, i.e.,

$$(\forall t \in D)(M\text{-of-}V(t)=false),$$

(M-2) says that

$$(\forall t \in D)(M\text{-of-}U(t)=false \text{ AND } M\text{-of-}W(t)=false).$$

Obviously this means that deletion D against V should be translated into deletion D against both U and W. Because union is a set

operation, if U and W have the same tuple, the duplicates will be eliminated after union. However, the meaning of views approach ensures that no ambiguity happens in the translation. □

Rule D-3 (Deletion against a Difference View): "Suppose deletion D is issued against a difference view V=U-W, where U and W are union-compatible. Then D is translatable into update against either U or W or both if and only if SAP1 (see below) is solvable. The solution to SAP1 determines the translation." Justification: Suppose deletion D is issued against a difference view V=U-W, where U and W are union-compatible. The meaning of V was given by (M-3):

$(\forall t \in dom(V))(M\text{-of-}V(t)=(M\text{-of-}U(t)$ AND $\neg M\text{-of-}W(t)))$.

Then deletion D from V means the following:

$(\forall t \in D)(M\text{-of-}U(t)=false$ OR $M\text{-of-}W(t)=true)$.

This means that from a semantic point of view, there are three possible ways to realize the deletion of tuple t from V:

(Alt-1) Delete t from U.

(Alt-2) Insert t in W.

(Alt-3) Delete t from U and insert it in W.

Of course, arbitrary choice is unacceptable because it may result in a semantically inconsistent database state. Therefore, in order to realize deletion D against V, we must resolve this semantic ambiguity for every tuple t of D. We call this problem semantic ambiguity problem of type one (SAP1). Sometimes SAP1 is solvable using certain integrity constraints deduced from the view. For example, suppose EPM(ename, sex,age,...) is the entire employee relation and FEMP(ename, sex, age, ...) is the female employee relation. Then the difference view EMP-FEMP defines the male employee relation. It is clear that a deletion against EMP-FEMP must be translated into a deletion against EMP. In other words, (Alt-1) must always be chosen in this case.

However, there exists another case in which human interaction may be necessary to solve SAP1. For example, suppose SOCCER(ename,age,dept) is the relation of the employees who belong to the soccer club, while TENNIS(ename,age,dept) is the relation of the employees who belong to the tennis club. Then the difference view SOCCER-TENNIS defines the relation of the employees who belong to the soccer club but not tennis club. Now, suppose a deletion of employee e from the difference view is issued. Then which one of the three possible alternatives (Alt-1), (Alt-2) or (Alt-3) should be chosen? In fact, one of the following three different facts could be observed behind the deletion corresponding to the alternatives:

(Fact-1) Employee e is no longer a member of the soccer club.

(Fact-2) Employee e is now a member of the tennis club.

(Fact-3) Employee e is no longer a member of the soccer club and is now a member of the tennis club.

In general, it is impossible for the SAP1 solver to find a semantically correct solution without interacting with the user who issued the deletion. □

One comment is in order here. Dayal and Bernstein (9) impose a restriction that a view update should be translated into the same type of updates; for example, a deletion should be translated into deletion(s). But such a restriction is not reasonable from a semantic point of view. The above example shows that a deletion against a difference view may have to be translated into both a deletion and an insertion.

Rule D-4 (Deletion against a Projection View): "Suppose deletion D is issued against a projection view V=U[X], where X is a subset of att(R). Then D is always translatable into a deletion D1 against U, where

$D1=\{u \in U \mid \exists t \in D,\ u[X]=t\}$."

Justification: Suppose deletion D is issued against a projection view V=U[X], where X is a subset of att(U). Then, by (M-4), we obtain

$(\forall t \in D)(\forall u \in dom(U))(u[X]=t \Rightarrow$ $M\text{-of-}U(u)=false))$.

This implies that in order to delete D from U, we must delete every tuple u of U having t as its X value. □

Rule D-5 (Deletion against a θ-Restriction View): "Suppose deletion D is issued against a θ-restriction view V=U[XθY], where X and Y are θ-compatible. Then D is always translatable into the same deletion D against U."

Justification: Suppose deletion D is issued against a θ-projection view V=U[XθY], where X and Y are θ-compatible. Then $(\forall t \in D)$ $(M\text{-of-}V(t)=false)$ must hold. (M-5) implies

$(\forall t \in D)(M\text{-of-}U(t)=false$ OR $t[X] \neg \theta t[Y])$.

But, $t[X] \theta t[Y]$ is true for every t of D because D is assumed to be a subset of V. Therefore, we obtain

$(\forall t \in D)(M\text{-of-}U(t)=false)$.

This means that D itself becomes a deletion against U. This translation realized D against V. □

5.2.2 LOCAL TRANSLATION RULES FOR INSERTION

Rule I-1 (Insertion against a Direct Product View): "Suppose insertion I is issued against a direct product view V=UxW. Then I is translatable into insertions I1 and I2 against V

and W, respectively, if and only if the cross reference condition holds for the union VuI, where I1=I[att(U)]-U and I2=I[att(W)]-W."

Justification: Suppose insertion I is issued against a direct product view V=UxW, where I represents a set of tuples that the user wants to insert into V. Because the union VuI, that is the desired result of the insertion, must again be a direct product, the cross reference condition (refer to Rule D-1) must be satisfied:

$(\forall t,t' \in dom(V))(t,t' \in VuI \Rightarrow$
$t[dom(U)] \parallel t'[dom(W)] \in VuI).$

That is, whenever two tuples t and t' exist in VuI, the concatenation of the dom(U) part of t and the dom(W) part of t' must belong to VuI if it is a direct product. Now let us translate I into insertions I1 against U and I2 against W as follows (notice that (M-1) ensures that I can be translated into insertions again):

I1=I[att(U)]-U.
I2=I[att(W)]-W.

Then, VuI=(UuI1)x(WuI2). As with Rule D-1, the cross reference condition governs the translatability of an insertion against a direct product view. □

Rule I-2 (Insertion against a Union View): "Suppose insertion I is issued against a union view V=UuW, where U and W are union-compatible. Then I is translatable into insertions against either U or W or both if and only if SAP2 (see below) is solvable. The solution to SAP2 determines the translation."

Justification: Suppose insertion I is issued against a union view V=UuW, where U and W are union-compatible. Then the meaning of V given by (M-2) implies the following:

$(\forall t \in I)$(M-of-U(t)=true OR M-of-W(t)=true).

This means that, from a semantic point of view there are in general three possible ways to realize the insertion of tuple t into V:

(Alt-1) Insert t into U.
(Alt-2) Insert t into W.
(Alt-3) Insert t into both U and W.

Arbitrary choice is unacceptable because it may result in a semantically inconsistent database state. Therefore, in order to realize insertion I in V, we must resolve this semantic ambiguity for every tuple t of I. We call this problem semantic ambiguity problem of type two (SAP2). Similar arguments hold for SAP2 solvability as for SAP1. That is, SAP2 is sometimes solvable using certain integrity constraints deduced from the view. For example, suppose MEPM(ename,sex,age,...) is the male employee relation and FEMP(ename,sex,age,...) is the female employee relation. Then the union view MEMPuFEMP represents the entire employee relation. In this case, it is clear that an insertion against MEMPuFEMP is always translatable without any ambiguity whenever SAP2 solver detects the sex value of the insertion tuple, which suggests which translation alternative should be chosen.

However, there exists a situation in which human interaction may be required to solve SAP2. For example, consider the SOCCER(ename,age,dept) and TENNIS(ename, age,dept) relations again. Suppose an insertion of employee e is issued against the union view SOCCERuTENNIS. Then, in order not to result in any semantic ambiguity, we must identify which one of the following three facts exists behind the insertion:

(Fact-1) Employee e has become a member of the soccer club.
(Fact-2) Employee e has become a member of the tennis club.
(Fact-3) Employee e has become a member of both the soccer club and the tennis club.

The SAP2 solver may not be able to find a semantically correct solution without interacting with the user who issued an insertion against a union view. □

Rule I-3 (Insertion against a Difference View):"Suppose insertion I is issued against a difference view V=U-W, where U and W are union-compatible. Then I is always translatable into insertion I against U and deletion D against W, where D=I."

Justification: Suppose insertion I is issued against a difference view V=U-W, where U and W are union-compatible. Since the meaning of V is given by (M-3), insertion I against V means the following:

$(\forall t \in I)$(M-of-U(t)=true AND M-of-W(t)=false).

This suggests that I should be translated into insertion I against U and deletion D against W, where D=I as a set. It is easy to see that this translation realizes the desired insertion. As we mentioned in the context of Rule D-3, an insertion into a view is translated into another type of update, namely, a deletion. □

Rule I-4 (Insertion against a Projection View): "Suppose insertion I is issued against a projection view V=U[X], where X is a subset of att(U). Then I is translatable into insertion against U if and only if SAP3 (see below) is solvable. The solution to SAP3 determines the translation."

Justification: Suppose insertion I is issued against a projection view V=U[X],where X is a subset of att(U). By the meaning of a projection view given by (M-4), we must find a tuple u whose projection on X is t and which satisfies the meaning of U (M-of-U) in order to make it possible to insert a tuple t of I in V. Of course, such u must be unique for each t in order not to cause any semantic ambiguity.

Now, a problem, which we call the semantic ambiguity problem of type three (SAP3), may arise. As observed in SAP1 and SAP2, sometimes the problem can be solved by using certain integrity constraints extracted from the view definition. For example, SPC is defined as a projection view of the intermediate view V0.1(supplier, SP.part, PC.part, customer) on {supplier, SP.part, customer} in Figure 1 (a). In this case, because V0.1 has the time invariant property that SP.part value is always equal to PC.part value for each tuple, every insertion tuple (s,p,c) against SPC has a unique original image (s,p,p,c). In general, however, it may be difficult to solve SAP3 without human interaction. Null values may be used to fill the non-X values of u. For example, SQL/DS (7) allows this unless such attributes, i.e. non-X attributes, do not permit null values. However, this expediency must be used very carefully; otherwise the users will face the semantic ambiguity problem associated with the null values. □

Rule I-5 (Insertion against a θ-Restriction View): "Suppose insertion I is issued against a θ-restriction view V=U[XθY], where X and Y are θ-compatible. Then if t[X]θt[Y] holds for every tuple t of I, then I is translatable into insertion I1 against U, where I1=I."

Justification: Suppose insertion I is issued against a θ-restriction view V=U[XθY], where X and Y are θ-compatible. Because the meaning of V is defined by (M-5), every tuple t of I must satisfy the relationship t[X]θt[Y]. Otherwise the insertion will not be accepted. If it is satisfied, then I becomes an insertion against U. It is clear that the translated insertion realizes the desired insertion against V. □

5.3 UPDATE MODIFICATION RULES

As we mentioned in section 5.1, there is one exceptional case in which failure of a local translation does not immediately mean the failure of the view update. This is the case when the translation of an update against an intermediate view that satisfies the following conditions is attempted:

(Sit-1) The intermediate view is a direct product view.

(Sit-2) A θ-restriction view is defind on the intermediate view.

We will first discuss two modification rules; the deletion modification rule, and the insertion modification rule. Then the fourth semantic ambiguity problem will be discussed.

5.3.1 DELETION MODIFICATION RULE

Suppose a θ-restriction view V=U[XθY] is defined on level i and the intermediate view U on level i+1, on which V is defined, is a direct product view W1xW2, where W1 and W2 are on level i+2. Suppose D is a deletion against V. Since V is a θ-restriction view, D is immediately converted to a deletion against U by Rule D-5. Since U is a direct product view, the cross reference condition is checked for the difference U-D. Suppose the condition does not hold. This means that it is impossible to translate D into a lower level update (see Rule D-1). However, let us allow D to be modified to D' as follows:

(Con-1) Intersection of D' with V is equal to D.

(Con-2) For any tuple t in D'-D, either its W1 value is equal to the W1 value of some tuple in D, or its W2 value is equal to the W2 value of some tuple in D, i.e.,

$(\forall t \in D'-D)(t[att(W1)] \in D[att(W1)]$ OR $t[att(W2)] \in D[att(W2)])$.

(Con-3) The cross reference condition holds for U-D'.

Criteria Cri-1 and Cri-2 of section 4 are satisfied by Con-1 and Con-2, respectively. However, there may exist more than one such D', thereby conflicting with criterion Cri-3. A detailed discussion of this problem will be given in section 5.3.3. If a unique D' is found, however, by Con-3 it is possible to translate it into deletion(s) against W1 and W2 without violating the three updatability criteria. We call the above the deletion modification rule.

An example may help to clarify the above discussion. Consider view SPC in Figure 1. Suppose a deletion D0={(s1,p1,c1), (s1,p1,c2), (s3,p1,c1), (s3,p1,c2)} is issued against SPC. Then by Rule D-4, D0 is translated into deletion D0.1 (against the intermediate view V0.1), which is expressed as D0.1={(s1,p1,p1, c1), (s1,p1,p1,c2), (s3,p1,p1,c1), (s3,p1,p1, c2)}. Then by Rule D-5, D0.1 becomes a deletion against the intermediate view V0.1.1, which we denote as D0.1.1. Notice that the difference V0.1.1-D0.1.1 does not satisfy the cross reference condition. For example, because two tuples (s1,p1,p2,c2) and (s2,p2,p1,c1) are in it, the tuple (s1,p1,p1,c1), which is obtained by concatenating the supplier and part part of the first tuple and the part and customer part of the second tuple, must be in it (see the cross reference condition). But this is not true.

So we try to modify it according to the deletion modification rule. One possible modification is

D0.1.1'=D0.1.1u{(s1,p1,p2,c2),(s3,p1,p2,c2)}.

By Rule D-1, D0.1.1' is translatable into deletion {(s1,p1), (s3,p1)} against the base relation SP. It is easy to see that this deletion against the base relation realizes the deletion against SPC.

The remainder of this section examines why the notion of deletion modification does not work for situations other than the intermediate view which satisfies Sit-1 and Sit-2. First, consider a direct product view V(=UxW) defined on level i, where U and W are on level i+1. Suppose deletion D against V is translated into deletions D1 against W and D2 against W2, respectively. Now, suppose it is impossible to translate D1 into the next lower level. Can we modify it? The answer is no, because any modification conflicts with criterion Cri-1, that is, V may lose tuple(s) that must not be deleted. The same argument holds for the case in which V on level i is the projection view of the intermediate view U on X on level i+1, i.e. V=U[X]. Any modification of the deletion against U conflicts with Cri-1.

Second, consider a union view V(=UuW) defined on level i; that is, U and W are on level i+1. Suppose a deletion D against V is translated into deletions D1 against U and D2 against W, respectively, and suppose D1 cannot be translated into a lower level. From a non-semantic, i.e. operational, point of view, D1 in certain cases can be modified. From a semantic point of view, however, it can not be modified. In order to understand this, let us consider the following example: Suppose V=UuW and U=U1xU2, where U1={(a), (b)}, U2={(1), (2)}, W={(b,1), (c,1)} and deletion D={(a,1)} is issued against V (={(a,1), (a,2), (b,1), (b,2), (c,1)}). By Rule D-2, D is translated into deletion D1(=D) against U and D2(=D) against W. Then observe that the cross reference condition does not hold for U-D1= {(b,1), (a,2), (b,2)}. Since (b,1) exists in W, however, we can delete the same tuple from U-D1 (i.e., modify D1 to D1'=D1u{(b,1)}), without causing any additional deletions from V. Moreover, it becomes possible to translate it into deletion {(1)} against U2 which realizes the deletion against V. Notice that the cross reference condition holds for U-D1'. Therefore, modification is possible in the above sense, i.e., from the operational point of view. However, from a semantic point of view, this modification is unacceptable, because the two tuples (b,1) of U and (b,1) of W have different meanings. That is, (b,1) of U satisfies M-of-U and the same tuple of W satisfies M-of-W, which is different from M-of-U. Therefore, deleting (b,1) from U conflicts with criterion Cri-2. For this reason, we prohibit deletion modifications in this situation. The same argument holds for a difference view.

Third, let us consider the situation in which modification is allowed to an intermediate view which satisfies Sit-1 and Sit-2. Suppose V=U[XθY] and U=W1uW2. In this case, no modification is allowed, because is may conflict with Cri-2, that is, it may delete extraneous tuples from W1 and/or W2. The same argument holds for three other cases; (i) V=U[XθY] and U=W1-W2, (ii) V=U[XθY] and U=W[Z], and (iii) V=U[XθY] and U=W[ZθZ']. In the case where V=U[XθY] and U=W1xW2, we can see that the modification does not conflict with Cri-2 in the sense that the modification gives users a way to specify a deletion against W1 and/or W2 through view V. For example, suppose a user wants to delete {(a,1)} from W1 through view V, where W1(A,B)={(a,1), (b,2)}, W2(C,D)={(1,x),(2,y)}, U=W1xW2, and V=U[B=C]. Then the deletion modification rule is necessary. The user might issue deletion {(a,1,1,x)} against V. It is easy to see that deletion {(a,1,1,x)} against U, the result of translating the deletion against V by Rule D-5, cannot be translated into the intended deletion {(a,1)} against W1. Rather, we realize that the only way to make it possible is to modify it to the deletion {(a,1,1,x), (a,1,2,y)} against U whose modification does not conflict with the user's intention.

5.3.2 INSERTION MODIFICATION RULE

Modification of an insertion is allowed for exactly the same type of intermediate view as for deletion modification. A modification of I to I' must satisfy the following conditions:

(Con-1) The intersection of I' with V is equal to I.

(Con-2) For any tuple t in I'-I, either its W1 value is equal to the W1 value of some tuple in I, or its W2 value is equal to the W2 value of some tuple in I, i.e.,

$$(\forall t \in I'-I)(t[att(W1)] \in I[att(W1)] \text{ OR } t[att(W2)] \in I[att(W2)]).$$

(Con-3) The cross reference condition holds for UuI'.

We call this the insertion modification rule.

5.3.3 AMBIGUITY IN UPDATE MODIFICATION

Ambiguity exists in modifying deletions or insertions. Let us again take deletion D0 against view SPC that was used in section 5.3.1. In that case, deletion D0.1.1 against the intermediate view V0.1.1 was a candidate for modification. One modification of D0.1.1, to D0.1.1', was shown. Two other modifications, named D0.1.1" and D0.1.1'" are possible. These are listed below:

(Mod-1) D0.1.1'=D0.1.1u{(s1,p1,p2,c2),

(s3,p1,p2,c2)}.
(Mod-2) D0.1.1"=D0.1.1u{(s1,p1,p2,c2), (s2,p2,p1,c2), (s3,p2,p1,c1), (s3,p2,p1,c2), (s3,p3,p1,c1), (s3,p3,p1,c2)}.
(Mod-3) D0.1.1'"=D0.1.1'uD0.1.1".

The question is which alternative should be chosen. Arbitrary choice is unacceptable. Each alternative results in a different deletion against the base relations, as observed below:

(Ob-1) By Mod-1, deletion D0 against SPC will be realized by deleting {(s1,p1), (s3,p1)} from the base relation SP.

(Ob-2) By Mod-2, deletion D0 against SPC will be realized by deleting {(p1,c1), (p1,c2)} from the base relation PC.

(Ob-3) By Mod-3, deletion D0 against SPC will be realized by deleting {(s1,p1), (s3,p1)} from SP and {(p1,c1), (p1,c2)} from PC.

Notice that this ambiguity corresponds to the three alternatives mentioned in section 4. That is, we have the following choices:

(Choice-1) If D0 is issued to reflect the fact that suppliers s1 and s3 have stopped supplying part p1, then Mod-1 should be chosen.

(Choice-2) If D0 is issued to reflect the fact that customers c1 and c2 have stopped purchasing part p1, then Mod-2 should be chosen.

(Choice-3) If D0 is issued to reflect the above two facts, then Mod-3 should be chosen.

In order to avoid a semantically inconsistent database state, this ambiguity problem should be resolved. We call this the semantic ambiguity problem of type four (SAP4).

We should notice that solving SAP4 is essentially equivalent to solving the semantic ambiguity that may arise when an update is issued against a natural join view, because the update modification is only allowed when an update is on the intermediate view which satisfies (Sit-1) and (Sit-2). That is, in our framework, a natural join view V(A,B,C) of two relations U(A,B) and W(B,C) is defined as V=V0[A,U.B,C], where V0=U⋈W[U.B=W.B]. By Rule D-4 or Rule I-4, any update against the natural join view V is translatable into an update against V0 without causing any ambiguity. Therefore, ambiguity may arise when the update against V0 is translated into update(s) against U and/or W. This is exactly the situation that we discussed above. To address the semantic ambiguity problem of the natural join view update, Dayal and Bernstein (9,19) and Carlson and Arora (13) used functional dependencies; Bancilhon and Spyratos (14,15,16) introduced the concept of complementary views to supply extra semantic information which makes it possible to resolve the ambiguity in certain situations. However, since this problem ultimately requires interaction with users to capture the user's intention, their approaches are still not sufficient to construct a complete SAP4 solver. The action taken by SAP4 solver is essentially to ask which one of the possible modification alternatives the user prefers. If if can find a unique answer, then it can translate the update into lower level views without introducing any semantic ambiguity. Notice that the concept of update modification provides the universe of discourse for the solver to interact with the users.

5.4 VIEW UPDATE TRANSLATOR

This section supplements section 5.1. Our translator consists of five components; a translator body, and a total of four different types of ambiguity solvers for SAP1, SAP2, SAP3, and SAP4. The translator body has an interface to each of the four SAP solvers. The basic view support capability will be provided by the translator body itself. When no SAP solvers are available, it can only handle the view updates which are translatable by using Rules D-1, D-2, D-4, D-5, I-1, I-3, and I-5. If we could provide a SAP1 solver, then the view update support capability would be enhanced by Rule D-3. Similarly, if SAP2 and SAP3 solvers were available, then the capability would be enhanced by Rules I-2 and I-4, respectively. By providing a SAP4 solver, the translator could handle updates to natural join views. Of course, the capacity of each solver is also questioned. A poor solver would provide a poor view support capability. A rich one would provide a rich support. From the nature of the ambiguity problems, such solvers may involve the users in resolving the ambiguities. Otherwise, a good, i.e. a semantically correct, translation may not be obtained. However, the design of such solvers is an open problem.

A sample translation used throughout this paper could be handled by the translator body with a SAP4 solver: Recall that deletion D0 (see section 5.3.1) was issued against the view SPC (see Figure 1). Then the translator body translates D0 into D0.1 (see section 5.3.1) by Rules D-4. It translates D0.1 into D0.1.1 by Rules D-5. Then, it recognizes that D0.1.1 is on the intermediate view which satisfies (Sit-1) and (Sit-2) (see section 5.3). It calls the SAP4 solver to handle D0.1.1 translation. The SAP4 solver recognizes that the difference V0.1.1-D0.1.1 does not satisfy the cross reference condition, and begins to try modifying D0.1.1. In this case, there are three

possible modifications (see section 5.3.3). The problem of whether it can choose correct alternative depends on how good its problem solving capability is. The result is reported to the translator body. Depending on the answer, it proceeds with further translation. The SAP4 solver would interact with the user if necessary. If the translator body finds that the local translation is impossible, or if any ambiguity solver finds that it can not resolve ambiguity, the translator rejects the view update and notifies the user.

6.CONCLUSION.

This paper discussed the relational view update translation problem from a semantic point of view. Views were defined in terms of the relational algebra. The meanings of views were defined by using the time-varying first order predicate calculus. The view updatability criteria were re-examined from the semantic point of view. The mechanisms of our view update translator which can handle any view updates were presented. A total of ten local translation rules was deduced, each of which has a semantic basis. A deletion and an insertion modification rule were introduced to augment the translation capability provided by the local rules. Our view update translator consists of a translator body and four semantic ambiguity problem solvers of different types. The translation capability depends on the solvers available to the translator body and the problem solving capability they offer. From the nature of the ambiguities the solvers resolve, they may involve the users in resolving the ambiguities. Although, the four semantic ambiguity problems were explained, the design of such solvers is an open problem.

ACKNOWLEDGEMENT

Dr. Won Kim of IBM Research Laboratory, San Jose, California, read several versions of this paper and gave many constructive comments from various points of view. The author expresses his sincerest thanks to him for helping transform an incomprehensible paper into its present state.

REFERENCES

1) Codd,E.F.: Recent Investigations in a Relational Database System, Inf. Process. 74, pp. 1017-1021 (1974).

2) Chamberlin,D.D., Gray,J.N and Traiger,I.L.: Views, Authorization, and Locking in a Database System, Proc. AFIPS NCC, Vol. 44, pp. 425-430 (1975).

3) Stonebraker,M: Implementation of Integrity Constraints and Views by Query Modification, Proc. ACM SIGMOD, pp. 65-78 (1975).

4) Astrahan,M.M et.al.: System R: Relational Approach to Database Management, ACM TODS, Vol. 1, No. 2, pp. 97-137 (1976).

5) Fernandez,E.B., Summers,R.C. and Wood,C: Database Security and Integrity, (book), Addison-Wesley Pub. Co. (1981).

6) Held,G.D., Stonebraker,M. and Wong,E.: INGRES - A Relational Data Base System, Proc. AFIPS NCC, Vol. 44, pp. 409-416 (1975).

7) SQL/DS System Concepts and Facilities, GH24-5013-0, File No. S370-50, IBM (1981).

8) Paolini,P. and Pelagatti,G.: Formal Definition of Mappings in a Database, Proc. ACM SIGMOD, pp. 40-46 (1977).

9) Dayal,U. and Bernstein, P.A. : On the Updatability of Relational Views, Proc. VLDB, pp. 368-377 (1978).

10) Furtado,A.L., Sevic,K.C. and dos Santos,C.S. : Permitting Updates Through Views of Data Bases, Inform. Systems, Vol. 4, No. 4, pp. 269-283 (1979).

11) Osman,I.M.: Updating Defined Relations, Proc. AFIPS NCC, Vol. 48, pp. 733-740 (1979).

12) Codd,E.F.: A Relational Model for Large Shared Data Banks, Comm. ACM, Vol. 13, No.6, pp. 909-917 (1970).

13) Carlson,C.R. and Arora,A.K.: The Updatabilty of Relational Views Based on Functional Dependencies, Proc. COMPSAC, pp. 415-420 (1979).

14) Bancilhon,F.: Supporting View Updates in Relational Data Base, Proc. IFIP TC-2 Working Conf. on Data Base Architecture, pp. 198-219 (1979) .

15) Spyratos,N.: Translation Structure of Relational Views, Proc. VLDB, pp. 411-416 (1980).

16) Bancilhon,F. and Spyratos,N.: Update Semantics of Relational Views, ACM TODS, Vol. 6, NO. 4, pp. 557-575 (1981).

17) Codd,E.F.: Relational Completeness of Database Sublanguages, in Data Base Systems, Courant Computer Sci. Symp. 6, Rustin,R. ed., Prentice-Hall, Englewood Cliffs, pp. 65-97 (1972).

18) Keller,A.M. : Updates to Relational Database Through Views Involving Joins, IBM Res. Rep. RJ3282, (October 1981).

19) Dayal,U. and Bernstein,P.A.: On the Correct Translation of Update Operations on Relational Views, ACM TODS, Vol. 7, No. 3, pp. 381-416 (1982).