USER-VIEW UPDATES IN RELATIONAL DATABASE SYSTEMS - A SEMANTIC APPROACH

Stephen J. Hegner
Department of Computer Science
University of Mississippi
University, Mississippi 38677

(March, 1979)

To appear in the Proceedings of the 1979 Conference on Information Sciences and Systems, The Johns Hopkins University, 28-30 March.

USER-VIEW UPDATES IN RELATIONAL DATABASE SYSTEMS - A SEMANTIC APPROACH

Stephen J. Hegner
Department of Computer Science
University of Mississippi
University, Mississippi 38677

Abstract
The problem of supporting updates in user views of a relational database system is investigated. The entire system is modelled within the framework of many-sorted first-order logic. Updates are then viewed as changes in the information (or valid sentences) of the system. From this point of view, it becomes clear which updates should be allowed.

1. Introduction to the Problem

For a variety of reasons, it is often desirable that different users of a database system have different views of the data. In this paper, the suitability of the relational model as the basis of a multi-user system is investigated from the point of view of user updates.

The relational model, originally conceived by Codd [2], has been widely documented (see e.g. [4]), and familiarity with it shall be assumed. Shown below is a simple relational database for an operation which deals with parts, suppliers, and projects which use these parts.

PARTNUMBER	PARTNAME	WEIGHT	
101	Widget	10	
102	Framoid	15	Rel.1
105	Thing	8	
109	X-J	100	

SUPPLIER	PARTNUMBER
Acme	101
Best	101
Acme	102
Acme	105

Rel.2

PARTNUMBER	PROJECT
101	Alpha
101	Beta
102	Gamma

Rel.3

The above is the base or system view of the data, and represents the actual data in the system. A user view is a functional image of the base under simple relational operations [3]. For example, user A may only be allowed to view columns 1 and 3 of Rel.1, so he sees

PARTNUMBER	WEIGHT	
101	10	
102	15	Rel.A
105	8	
109	100	

as his view. This is the {1,3}-projection of Rel.1. User B might wish to view the join of Rel.2 and Rel.3; his view would then be

SUPPLIER	PARTNUMBER	PROJECT	
Acme	101	Alpha	
Acme	101	Beta	
Best	101	Alpha	Rel.B
Best	101	Beta	
Acme	102	Gamma	

There is no conceptual problem in constructing these views. Consider, however, what happens when the users are allowed to alter the database through these views. Suppose user A wishes to insert (181,3) into Rel.A. To achieve this, (181,?,3) must be inserted into Rel.1, where ? is some unknown quantity. Insertion of this tuple requires either an arbitrary value to be used for ?, or else to specify that it is "null" or unknown. The former is arbitrary and completely unacceptable. The latter has been proposed by some, but can lead to problems when operations such as join are later applied.

An even more serious update problem has been illustrated by Codd [3]. Suppose user B wishes to delete (Acme,101,Alpha) from his view. This requires the deletion of either (Acme,101) from Rel.1 or (101,Alpha) from Rel.2. If the former is deleted, (Acme,101,Beta) will also be deleted from the user's view, and if the latter is deleted, (Best,101,Beta) will also be deleted from the user's view. Similar problems occur if user B attempts to insert, for example, (A1,105,Gamma) into Rel.B.

The above two examples clearly illustrate that user-view updates must be handled with care. While others have taken a syntactic approach to this problem [6], the approach presented here takes a semantic direction in which all updates are regarded as changes in the information content of the database.

2. The Logical Model

The framework of the approach to the syntax and semantics of relational databases which is presented here is a structural variation of first-order many-sorted logic. The principal difference between the usual (one-sorted) logic and many-sorted logic is that in the latter, more than one universe is allowed. This is useful in database theory since the logic concept of universe of discourse corresponds to the database concept of domain.

It is necessary to be terse in the defintion of

the basic concepts. However, examples are provided to increase clarity. When a question arises, a reference to any good textbook on mathematical logic, such as [8], should clear things up.

DEFINITION A relational database schema is a 5-tuple $L=(S,\mathcal{D},F,R,T)$, where
(a) S is a finite nonempty set, the set of sorts.
(b) $\mathcal{D}$ is a finite nonempty set of finite sets, the set of domains.
(c) $F:S\to\mathcal{D}$ is a function. For each $S\in S$, $F(S)$ is called the set of constants of sort S.
(d) R is a nonempty set, the set of relation symbols.
(e) $T:R\to S^+$ is a function (where S^+ is the set of all finite nonempty strings in S). For $R\in R$, $T(R)$ is the rank of R.

As an example, let S={PARTNUMBER,PARTNAME, WEIGHT,SUPPLIER,PROJECT}, $\mathcal{D}$={{0,1,..,1000}, {Widget,Framoid,Thing,X-J,KP-7},{Acme,Best,Ace, Al,Smith},{Alpha,Beta,Gamma,Delta,Epsilon}}; F(PARTNUMBER)={0,1,..,1000}, F(PARTNAME)={Widget, Framoid,Thing,X-J,KP-7}, F(WEIGHT)={0,1,..,1000}, F(SUPPLIER)={Acme,Best,Ace,Al,Smith}, F(PROJECT) ={Alpha,Beta,Gamma,Delta,Epsilon}; R={R1,R2,R3, R4,R5}, T(R1)=(PARTNUMBER,PARTNAME,WEIGHT), T(R2) =(SUPPLIER,PARTNUMBER), T(R3)=(PARTNUMBER,PRO-JECT), T(R4)=(PARTNUMBER,WEIGHT), T(R5)= (SUPPLIER,PARTNUMBER,PROJECT).

DEFINITION Let $L=(S,\mathcal{D},F,R,T)$ be a relational database schema.
(a) A partial interpretation schema of L is a subset Q of R. A total interpretation schema is just R.
(b) Let Q be a partial interpretation schema of L. A Q-interpretation of L is a function I which assigns to each $R\in Q$ a relation $R^I\subseteq F(S1)\times..\times F(Sn)$, where $T(R)$=S1,..,Sn. An R-interpretation is also called a total interpretation.

As an example, let B={R1,R2,R3}, U1={R4}, U2={R5}. Each is a partial interpretation schema. Let I be defined on B by Ri↦Rel.i for $1\le i\le 3$, I1 be defined on U1 by R4↦Rel.A, and I2 be defined on U2 by R5↦Rel.B, where the Rel.x's are from the introduction. Then I (resp. I1, resp. I2) is a B- (resp. U1-, resp. U2-) interpretation of L.

The above framework is lacking in that there is no way to express constraints on the data. To do so requires the introduction of axioms. Let $L=(S,\mathcal{D},F,R,T)$ be a relational database schema. For each $S\in S$, introduce a countable set V^S of variables of sort S. A term of sort S is either a variable of sort S or a constant of sort S. An atomic well-formed formula (atomic wf) over L is of the form R(t1,..,tn), where $R\in R$, rank(R)=S1,..,Sn, and ti is a term of sort Si for $1\le i\le n$. A wf over L is either an atomic wf, is of the form t1=t2 where t1 and t2 are terms of the same sort, or is built up in the usual way from other wf's and variables using $\neg,\wedge,\vee,\Rightarrow,\Leftrightarrow,\forall,\exists$. A sentence is a wf in which all variables are bound (quantified). Given a sentence A over L and a total interpretation I of L, the usual and obvious meaning is given to "A is satisfied in I". Given a partial interpretation schema Q of L and a Q-interpretation J, A is satisfied in J if there is a total interpretation I in which A is satisfied which is an extension of J.

Constraints on the database are expressed by sentences, and may be broken into two classes. First of all, there are semantic integrity constraints, which express constraints made upon the data based upon properties of the reality which is being modelled. Examples are functional, multivalued and mutual dependencies. It has already been shown that such constraints can be expressed using sentences in a first-order logic [9]. Second, there are consistency constraints which specify redundancy within the data. In the example, the constraints that R4 is the $\{1,3\}$-projection of R1 and that R5 is the join of R2 and R3 are such constraints, and are expressed by the following sentences:

$$(\forall x)(\forall y)(R4(x,y)\Leftrightarrow(\exists z)R1(x,z,y));$$
$$(\forall x)(\forall y)(\forall z)(R5(x,y,z)\Leftrightarrow R2(x,y)\wedge R3(y,z)). \quad (1)$$

It is a fundamental hypothesis of this paper that all consistency constraints can be so expressed. It is a simple matter to verify that this is indeed the case for join, projection, composition, and permutation, and motivates the following definition.

DEFINITION A relational database schema with semantics is a 7-tuple $L=(S,\mathcal{D},F,R,T,A_i,A_c)$, where $(S,\mathcal{D},F,R,T)$ is a relational database schema and A_i and A_c are independent sets of sentences, the integrity axioms and consistency axioms, respectively.

Recall that two sets of sentences A and B are independent if no element of B can be deduced from A, and vice-versa. The assumption that the integrity axioms are independent of the consistency axoims is necessary so that the two may always be distinguished, and is always met in practice.

DEFINITION Let $L=(S,\mathcal{D},F,R,T,A_i,A_c)$ be a relational database schema with semantics.
(a) A total model (or just model) of L is a total interpretation I of $(S,\mathcal{D},F,R,T)$ such that each element of $A_i\cup A_c$ is satisfied in I.
(b) A partial interpretation schema of L is just a partial interpretation schema Q of $(S,\mathcal{D},F,R,T)$. A Q-model of L is a Q-interpretation J of $(S,\mathcal{D},F,R,T)$ which extends to a total model of L.

In the example, let A_c be the sentences (1) above. Let A_i consist of

$$(\forall v)(\forall x)(\forall y)(\forall z)(\forall w)(R1(x,y,z)\wedge R1(x,w,v)\Rightarrow y=w); \quad (2)$$
$$(\forall v)(\forall x)(\forall y)(\forall z)(\forall w)(R1(x,y,z)\wedge R1(x,v,w)\Rightarrow z=w).$$

These express the fact that the functional dependencies 1→2 and 1→3 must hold for R1. The relations in the example satisfy both sets of axioms.

It is now possible to develop the concept of a multi-user relational database system.

DEFINITION Let $L=(S,\mathcal{D},F,R,T,A_{\mathcal{L}},A_C)$ be a relational database schema with semantics, and let Q be a partial interpretation schema of L.
(a) The closure of Q, denoted $Cl(Q)$, is the maximal $P \leq R$ such that every Q-model extends to a unique P-model of L.
(b) Q is dense if $Cl(Q)=R$.
(c) Given another partial interpretation schema O of L, Q determines O if $Cl(O) \leq Cl(Q)$.
(d) The data span (resp. information span) of Q, denoted D-span(Q) (resp. I-span(Q)), is the set of all sentences over L which involve only those relation symbols from Q (resp. $Cl(Q)$).

DEFINITION A multi-user relational database schema is a triple $M=(L,B,U)$, where $L=(S,\mathcal{D},F,R,T,A_{\mathcal{L}},A_C)$ is a relational database schema with semantics, B is a partial interpretation schema which is dense, and $U=\{U1,..,Un\}$ is a set of partial interpretation schemata such that $\{B,U1,..,Un\}$ is a partition of R. $\{B\} \cup U$ is called the set of views of M. B is the base view schema; the Ui's are called the user view schemata. A state of M is just a model of L. Given a state I, the restriction of I to B (resp. Ui) is called the base state or B-state (resp. ith user state or Ui-state).

Note that by the definition of dense, the base state uniquely determines each user state in any interpretation. Also, the restriction that the views be pairwise disjoint is no problem, since things can be set up so that different relation symbols always give rise to the same relations in an interpretation.

In the example, $B=\{R1,R2,R3\}$, $U=\{U1,U2\}$, where $U1=\{R4\}$, $U2=\{R5\}$.

3. Information, Data, and the Update Strategy

For the rest of this paper, fix a multi-user relational database schema $M=(L,B,U)$ with $L=(S,\mathcal{D},F,R,T,A_{\mathcal{L}},A_C)$.

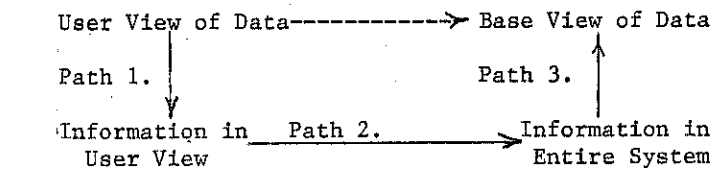
DEFINITION Let Q be a view of M, and let I be a state of Q.
(a) The information of the corresponding Q-state, denoted Info(I,Q), is the set of all sentences in I-Span(Q) which are satisfied in I.
(b) The data of the corresponding Q-state, denoted Data(I,Q), is the set of all relations R^I with $R \in Q$.

The distinction made in the previous defintion must not be underemphasized. Data are sets of tuples, while information is sentences. The data in an interpretation are a representation of the information of that interpretation. It is this distinction, which is clouded in most approaches to database systems, which forms the cornerstone of the approach developed here. It is, nonetheless, necessary to be able to relate information and data. This link is now developed.

DEFINITION Let $\mathcal{B}$ be a set of sentences over L.
(a) A pure consequence of $\mathcal{B}$ is a sentence A over L which is true in any total interpretation of $(S,\mathcal{D},F,R,T)$ in which each element of $\mathcal{B}$ is true. Note that the axioms of L may not be used to restrict consideration to models. Write $\mathcal{B} \models A$ to denote that A is a pure consequence of $\mathcal{B}$, and write $Con(\mathcal{B})=\{A \mid \mathcal{B} \models A\}$.
(b) A basis for $\mathcal{B}$ is a set of sentences $\mathcal{C}$ such that $\mathcal{B}=Con(\mathcal{C})$.
(c) $\mathcal{B}$ is total with respect to $Q \leq R$ if for each sentence $A \in Cl(Q)$, $A \in \mathcal{B}$ or $\neg A \in \mathcal{B}$.
(d) A sentence A over L is elemental if it is an atomic wf or the negation of an atomic wf. $\mathcal{B}$ is data-determining if it is total and has a basis consisting of elemental sentences.

It is important to know when $\mathcal{B}=Info(I,Q)$ for a given model I. This is easily answered, as $\mathcal{B}=Info(I,Q)$ if and only if $\mathcal{B}$ is data-determining, Since $R(t1,..,tn) \in Info(I,Q)$ if and only if $(t1,..,tn) \in R^I$.

The fundamental update principle is that all updates are processed as information updates. This principle is presented schematically as follows.



The dotted path is traversed by chasing the solid paths in order. The idea is to determine what the update semantics (information) are and to base any data manipulation upon this information update. An outline of the procedure used to implement this principle is presented next. In the procedure, there are two variables, $\mathcal{I}$ and $\mathcal{J}$. Each takes as values sets of sentences over L, and each is initially empty. Assume that the current state of the system is I.

Step 1 (Path 1): From the specified data update, determine the corresponding information of that view which is to be changed. Insert this information into $\mathcal{J}$.

Step 2 (Path 2): From the consistency constraints, determine the primary (base view) information updates; insert into $\mathcal{I}$.

Step 3a: Check to see if the updates has a basis of elemental sentences. If so, set $\mathcal{I}$ to this basis (which is unique). Otherwise, the update cannot be made.

Step 3b: Check to see if the update $\mathcal{I}$ will satisfy the integrity constraints. If not, the update cannot be made.

Step 3c: Determine any secondary (user view) information updates from $\mathcal{I}$; insert this information

into J.

Step 3d (Path 3): Perform the data update corresponding to $I \cup J$.

Only three very simple classes of updates will be considered. They are of the following forms:
(a) Insert T into R;
(b) Delete T from R;
(c) Replace T1 with T2 in R.
In each case $R \in Ui$ for some i, and T, T1, and T2 are tuples of the appropriate rank. It shall also be assumed that each specified update is nontrivial, in the sense that a tuple to be inserted is not already present, a tuple to be deleted is present, and a tuple to be replaced is present while its replacement is not, and the two are distinct. These requirements, as well as that of processing only one update at a time, are imposed only in the interests of simplicity. The generalizations to eliminate these restrictions are not conceptually difficult.

4. Nature of the Constraints

Before expanding upon the steps of the procedure, it is necessary to consider the constraints in some detail. If arbitrary sentences are allowed as integrity and consistency axioms, the handling of updates becomes extremely difficult. In this section, standard forms for these axioms are specified which make update handling tractable, and yet which cover the cases found in practice.

ASSUMPTION In the schema M, all axioms in A_i shall be in D-span(B).

This means that all integrity axioms shall be specified on the base view. Since the base view determines each user view, this is not a compromising restriction. However, it will allow a more compact checking of update integrity. The form in which the consistency axioms will be placed is somewhat more restrictive.

DEFINITION (a) A primary atom is a wf of the form R(t1,..,tn), with $R \in B$ and t1,..,tn variables.
(b) The primary clauses are defined by the following:
(i) A primary atom is a primary clause;
(ii) If C1,..,Cn are primary atoms, $C1 \wedge .. \wedge Cn$ is a primary clause.
(iii) If C is a primary clause and x is a variable free in C, then $(\exists x)C$ is a primary clause.
(c) A secondary clause is an atom of the form R(t1,..,tn), with $R \in Ui$ for some i and t1,..,tn variables.
(d) A consistency sentence over M is of the form $(\forall x1)..(\forall xk)(Cu \Leftrightarrow Cb)$, where Cu is a secondary clause and Cb is a primary clause, each with the same free variables, and x1,..,xk are the variables in Cu in the order in which they occur. The sentence is said to be a consistency sentence for R when Cu=R(t1,..,tn).

ASSUMPTION In the schema M, all axioms in A_c shall be consistency sentences over M. Furthermore, there shall be exactly one axiom for each R $\in U$.

This means that the only consistency constraints are those imposed by the base view on the user views. This restricted form of consistency sentences is justified, because all user view operations used in database systems can be described using such constraints. Examples of join and projection have already been given by sentences (1). Simple examples of composition and permutation constraints are given below.

$(\forall x)(\forall y)(T(x,y) \Leftrightarrow (\exists z)(R(x,z) \wedge T(z,y)))$;
$(\forall x)(\forall y)(T(x,y) \Leftrightarrow R(y,x))$.

5. Expansion of the Procedure

In the following, each of the six steps of the procedure for processing an update shall be considered in turn. In this short report, the main idea of each step shall be explained, and then a series of examples illustrating the step shall be considered. A completely detailed algorithm will appear in a forthcoming full report.

There are five update examples which shall be (independently) treated in the following. They are applied to the relational database example developed earlier in this paper, and are subsequently referenced by number as follows:
(1) Delete (101,10) from R4;
(2) Insert (181,3) into R4;
(3) Insert (A1,105,Gamma) into R5;
(4) Delete (Best,101,Alpha) from R5;
(5) Replace (Acme,101,Alpha) with (Acme,101,Epsilon) in R5.

Expansion of Step 1

In this step, the data changes in the user-specified update is translated to an information update, without regard to any system axioms. This step is extremely simple; the values assigned to J are as follows.

(a) If the update is insert T into R, set $J=\{R(T)\}$
(b) If the update is delete T from R, set $J=\{\neg R(T)\}$.
(c) If the update is replace T1 with T2 in R, set $J=\{\neg R(T1), R(T2)\}$.

J is given below for each of the five examples.

(1) $J=\{\neg R4(101,10)\}$.
(2) $J=\{R4(181,3)\}$.
(3) $J=\{R5(A1,105,Gamma)\}$.
(4) $J=\{\neg R5(Best,101,Beta)\}$.
(5) $J=\{\neg R5(Acme,101,Alpha), R5(Acme,101,Epsilon)\}$.

Expansion of Step 2

In this step, it must be determined which information updates in the base view are implied by the information in J determined in step 1. With general consistency axioms, this becomes very difficult, but the use of consistency sentences makes the process quite straightforward. Each elemental sentence in J is bound to the consistency constraint in which it occurs. Variables

in the constraint are bound to the corresponding constants in the elemental sentence. The result is a primary clause or its negation with its free variables bound to constants. It is this resulting primary sentence which is put into I. The examples should make this clear.

(1) It is necessary to bind $\neg$R4(101,10) to its consistency constraint W, which is $(\forall x)(\forall y)(R4(x,y) \Leftrightarrow (\exists z)R1(x,z,y))$. First, R4(101,10) is bound to this constraint to get $(\exists z)$R1(101,z,10). Then, since the original sentence $\neg$R4(101,10) is a negated atom, the negation of the binding of R4(101,10) to W is placed in I to yield $I=\{\neg(\forall z)R4(101,x,10)\}$.

The other four examples proceed in an essentially similar fashion. The results are given below.
(2) $I=\{(\exists z)R1(181,z,3)\}$.
(3) $I=\{$R2(A1,105) R3(105,Gamma)$\}$.
(4) $I=\{\neg(R2(Best,101)\wedge R3(101,Beta))\}$.
(5) $I=\{\neg(R2(Acme,101)\wedge R3(101,Alpha)),$
$(R2(Acme,101)\wedge R2(101,Epsilon))\}$.

Expansion of Step 3a
The idea of the information update is to take the sentences in I and update the information in Info(I,B) with I. Now Info(I,B) is data determining by definition, and so it has a basis consisting of elemental sentences. The updated information must also be data determining, in order that it determine a new state for the database. Clearly, if I has a basis consisting of elemental sentences, it is a simple matter to perform the information update; the new information for the base system is just Info(I,B) with the elements of I added in and their negations deleted. This new information will clearly be data determining. While it is possible to perform updates resulting in data determining information even when I does not have an elementary basis, this case connot occur within the update framework presented here and so will not be considered. The examples are handled as follows.

(1) The only sentence in I is $\neg(\exists z)$R1(101,x,10), which is the same as $(\forall z)(\neg$R1(101,x,10). This has the elemental base $\{\neg$R1(101,Widget,10), $\neg$R1(101,Framoid,10),$\neg$R1(101,Thing,10), $\neg$R1(101,X-J,10),$\neg$R1(101,KP-7,10)$\}$. Note that the conversion required the use of all elements from the PARTNAME domain, and that the fact that this domain is finite is crucial.
(2) The only sentence in I is $(\exists z)$R1(181,z,3). It has no elemental base. In general, unless the domain of existential quantification has only one element (a trivial case), there is no way to convert an existentially quantified primary clause to elements.
(3) The only sentence in I is R2(A1,105) R3(105,Gamma), which clearly has the elemental basis $\{$R2(A1,105),R3(105,Gamma)$\}$.
(4) The only sentence in I is $\neg$(R2(Best,101)$\wedge$ R3(101,Beta)). It has no elemental basis.
(5) The sentences in I are $\neg$((R2(Acme,101)$\wedge$ R3(101,Alpha)) and R2(Acme,101)$\wedge$R2(101,Epsilon). The first is equivalent to $(\neg$R2(Acme,101))$\vee$ $(\neg$R3(101,Alpha)). However, the second implies that R2(Acme,101) is true, so $\neg$R3(101,Alpha) must be true. Hence I has the elemental base $\{\neg$R3(101,Alpha),R2(Acme,101),R3(101,Epsilon)$\}$.

Expansion of Step 3b
Checking to see if the specified updates satisfy the integrity constraints of the system is just a matter of checking whether or not the information update implied by I will satisfy the integrity constraints. The difficulty of this task depends upon the formulation of the integrity constraints. In general, quite sophisticated theorem-proving techniques are required. However, for the cases of functional and multivalued dependencies, complete sets of inference rules for deriving all such dependencies from a given base are known [1,7]. In such cases the checking is a straightforward test. This step of the procedure is at any rate common to any approach to updates, and shall not be considered further here. All of the examples which have survived this far satisfy the integrity constraints (2) specified earlier.

Expansion of Step 3c
In this step, the information update I for the base view must be extended to the user views. In the example system, the views are noninteracting, so the effects will only be on the user views which originated the update. In general, however, it must be remembered that an update from one view may affect another view. The process is essentially similar to that of step 2, except that this time the consistency constraints are bound to primary clauses. The examples which have survived this far are now presented.
(1) Using the constraint $(\forall x)(\forall y)(R4(x,y) \Leftrightarrow (\exists z)R1(x,z,y))$, the elements of I are in turn bound to the right-hand side to give $J=\{\neg R4(101,10)\}$, exactly the previous J.
(3) Using the constraint $(\forall x)(\forall y)(\forall z)(R5(x,y,z) \Leftrightarrow R2(x,y)\wedge R3(y,z)$, binding I to the right-hand side yields $\{$R5(A1,105,Gamma), R5(Acme,105,Gamma)$\}$. Note that the binding required reference to Info(I,B), to determine the satisfiability of the primary clause.
(5) Using the same constraint as in (3), binding I to the right-hand side yields $J=$ $\{\neg$R5(Acme,101,Alpha),$\neg$R5(Best,101,Alpha), R5(Acme,101,Epsilon),R5(Best,101,Epsilon)$\}$.

Expansion of Step 3d
Using the correspondence between the elemental sentences and data, this step is trivial. The examples are as follows:
(1) Delete (101,Widget,10) from Rel.1 and delete (101,10) from Rel.A.
(3) Insert (A1,105) into Rel.2, (105,Gamma) into Rel.3, and (A1,105,Gamma) and (Acme,105,Gamma) into Rel.B.
(5) Delete (101,Alpha) from Rel.3; insert (101,Epsilon) into Rel.3; delete (Acme,101,Alpha) and (Best,101,Alpha) from R5; insert (Acme,101,Epsilon) and (Best,101,Epsilon) into R5.

6. Observations and Conclusions

It is quite pertinent to examine more closely a few of the update examples. Consider first the join. Insertion of (A1,105,Gamma) was permitted under this approach, even though it led to insertion of the side effect (Acme,105,Gamma). However, deletion of (Acme,101,Alpha) was not permitted. This asymmetry may at first seem to be a defect in the approach, but upon closer examination, it is quite reasonable. From an information point of view, insert (A1,105,Gamma) means that R5(A1,105,Gamma) is now true, which is the same as both R2(A1,105) and R3(105,Gamma). In words, the user has said that supplier A1 now supplies part 105 and part 105 is used on project Gamma. Since R5(Acme,105,Gamma) says that Acme supplies part 105 and part 105 is used on project Gamma, it it stands to reason that this should appear in the user's view, since it is now true. On the other hand, when the user specifies the deletion of (Acme,101,Alpha), he is saying that R5(Acme,101,Alpha) is now false. This means that R2(Acme,101) or R3(101,Alpha) is false. Unless the user further specifies which of the two is false, the update cannot be processed, since there is no way to represent the fact that Acme does not supply part 101 or part 101 is no longer used by project Alpha in the database.

Consider next the insertion of (181,3) into the projection of Rel.1. The information to be inserted is $(\exists x)$R1(181,x,3). Because it does not have an elemental base, it cannot be inserted according to the rules laid down. Nonetheless, it has been widely proposed that this may be handled by inserting a "blank" in the second field of Rel.1. However, if a new view is subsequently defined in which the second field of Rel.1 is used in a join (for example), problems in supporting this new view will clearly be encountered.

As a general rule, deletion from joins and compositions and insertions into compositions and projections are not possible with a relational base view when the semantic update approach is used. The semantic approach to updates does not regard side effects per se as reasons for rejecting an update, as the side effects may very well be logical consequences of the specified update. Rather, it is the impossibility of the information of a data update in the user's view to be reflected as a data update in the base view which causes an update to fail, because the information change does not have an elemental basis. This suggests that another model of the base view, in which non-elemental sentences have a representation, might be more appropriate. In view of the approach to updates developed here, an obvious candidate is first-order logic itself. There is no reason that the base view need be data at all, as long as it is capable of supporting the user views. It seems likely that by using an information-oriented rather than data-oriented model as the base, a large variety of user views may be supported, including but not restricted to relational views. This is in the spirit of the coexistence model [10]. Already, at least one approach using predicate calculus has been forwarded [5]. However, the idea is yet in its infancy, and substantial results are not yet available.

References

1. Beeri, C., R. Fagin, and J. H. Howard, "A complete axiomatization for functional and multivalued dependencies in database relations", Proc. 1977 SIGMOD Conference, Toronto, ACM, pp. 41-61.
2. Codd, E. F., "A relational model of data for large shared data banks", Comm. ACM, 13,6(1970), pp. 377-387.
3. Codd, E. F., "Recent investigations in relational database systems", Proc. IFIP 74, North-Holland, pp.1017-1021.
4. Date, C. J., An Introduction to Database Systems, Second edition, Addison-Wesley, 1977.
5. Hafez, K., and T. G. Windeknecht, "A logical framework for database design", Proc. 1978 Information Sciences and Systems Conf., Baltimore, Johns Hopkins U., pp. 305-309.
6. Dayal, U., and P. A. Bernstein, "On the updatability of relational views", Proc. Fourth VLDB Conf., Berlin,1978, IEEE, pp. 368-377.
7. Mendelzon, A. O., "On axiomatizing multivalued dependencies in relational databases", J. ACM, 26,1(1979), pp. 37-44.
8. Monk, J. D., Mathematical Logic, Springer, 1976.
9. Nicolas, J. M., "First order logic formalization for functional, multivalued, and mutual dependencies", Proc. 1978 SIGMOD Conf., Austin, ACM, pp. 40-46.
10. Nijssen, G. M., "A gross architecture for the next generation database management systems", Modelling in Data Base Management Systems, ed. by G. M. Nijssen, North-Holland, 1976, pp. 1-24.

Acknowledgments

The author's work on the user-view update problem began as a joint effort with Ruth Maulucci; this early collaboration is gratefully acknowledged. The author also wishes to thank Khaled Hafez for several enlightening discussions on the role of logic in database design.