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)

Abstract
The problem of supporting updates in user views of a relational database system is investigated. The entire system is modeled 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 products which use these parts.

<table>
<thead>
<tr>
<th>PARTNUMBER</th>
<th>PARTNAME</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Widget</td>
<td>10</td>
</tr>
<tr>
<td>102</td>
<td>Framotl</td>
<td>15</td>
</tr>
<tr>
<td>105</td>
<td>Thing</td>
<td>8</td>
</tr>
<tr>
<td>109</td>
<td>X-J</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUPPLIER</th>
<th>PARTNUMBER</th>
<th>PARTNUMBER</th>
<th>PROJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acme</td>
<td>101</td>
<td>101</td>
<td>Alpha</td>
</tr>
<tr>
<td>Acme</td>
<td>101</td>
<td>101</td>
<td>Beta</td>
</tr>
<tr>
<td>Best</td>
<td>101</td>
<td>101</td>
<td>Alpha</td>
</tr>
<tr>
<td>Best</td>
<td>101</td>
<td>101</td>
<td>Beta</td>
</tr>
<tr>
<td>Acme</td>
<td>102</td>
<td>101</td>
<td>Gamma</td>
</tr>
<tr>
<td>Acme</td>
<td>105</td>
<td>101</td>
<td>Gamma</td>
</tr>
</tbody>
</table>

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, (AI,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 [5], 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 definition 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 \( l=(S,D,F,R,T) \), where

(a) \( S \) is a finite nonempty set, the set of sorts.
(b) \( D \) is a finite nonempty set of finite sets, the set of domains.
(c) \( F:S \times D \) 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) \( T:R \times S^+ \) is a function (where \( S^+ \) is the set of all finite nonempty strings in \( S \)). For \( R \subseteq \mathbb{R} \), \( T(R) \) is the rank of \( R \).

As an example, let \( S=\{ \text{PARTNUMBER}, \text{PARTNAME}, \text{WEIGHT}, \text{SUPPLIER}, \text{PROJECT} \} \), \( D=\{0,1,\ldots,1000\} \), \{Framoid, Framoid, Thing, X-J, KP-7\}, \{Acme, Best, Ace, Al, Smith\}, \{Alpha, Beta, Gamma, Delta, Epsilon\}; \( F(\text{PARTNUMBER})=\{0,1,\ldots,1000\} \), \( F(\text{PARTNAME})=\{\text{Framoid, Framoid, Thing, X-J, KP-7}\} \), \( F(\text{WEIGHT})=\{0,1,\ldots,1000\} \), \( F(\text{SUPPLIER})=\{\text{Acme, Best, Ace, Al, Smith}\} \), \( F(\text{PROJECT})=\{\text{Alpha, Beta, Gamma, Delta, Epsilon}\} \), \( R=(R_1, R_2, R_3, R_4, R_5) \), \( T(R_1)=\{\text{PARTNUMBER}, \text{PARTNAME}, \text{WEIGHT}\} \), \( T(R_2)=\{\text{SUPPLIER}, \text{PARTNUMBER}\} \), \( T(R_3)=\{\text{PARTNUMBER}, \text{PARTNAME}, \text{SUPPLIER}, \text{PROJECT}\} \), \( T(R_4)=\{\text{PARTNUMBER}, \text{PARTNAME}, \text{WEIGHT}\} \), \( T(R_5)=\{\text{SUPPLIER}, \text{PARTNUMBER}, \text{PROJECT}\} \).

**DEFINITION** Let \( l=(S,D,F,R,T) \) 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 \( R \subseteq \mathbb{R} \) a relation \( R^Q_{S} \subseteq F(S) \times \ldots \times F(Sn) \), where \( T(R)=S_1, \ldots, S_n \). An \( R \)-interpretation is also called a total interpretation.

As an example, let \( B=(R_1, R_2, R_3) \), \( U_1=(R_4) \), \( U_2=(R_5) \). Each is a partial interpretation schema. Let \( I \) be defined on \( B \) by \( R_3=\text{Rel.1} \) for \( l \in l(S) \), \( R_1 \) is defined on \( U_1 \) by \( R_4=\text{Rel.2} \), and \( R_2 \) defined on \( U_2 \) by \( R_5=\text{Rel.3} \), where the \( \text{Rel.} \)'s are from the introduction. Then \( I \) (resp. \( I_1 \), resp. \( I_2 \)) is a \( B \)- (resp. \( U_1 \)-, resp. \( U_2 \)-) 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 \( l=(S,D,F,R,T) \) 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 \( \mathbf{w} \)) over \( l \) is of the form \( R(t_1, \ldots, t_n) \), where \( R \in \mathbb{R} \), \( \text{rank}(R)=S_1, \ldots, S_n \), and \( t_i \) is a term of sort \( S_i \) for \( \text{lg}(R) \). An \( \mathbf{w} \) over \( l \) is either an atomic \( \mathbf{w} \) of the form \( t_1=t_2 \) where \( t_1 \) and \( t_2 \) are terms of the same sort, or a \( \mathbf{w} \) in the usual way from other \( \mathbf{w} \)'s and variables using \( \neg, \land, \lor, \forall, \exists \). A sentence is a \( \mathbf{w} \) in which all variables are bound (quantified). Given a sentence \( A \) over \( l \) and a total interpretation \( I \) of \( l \), the usual and ob-

The text continues with the discussion of constraints on the database, expressed by sentences, and the introduction of additional concepts such as integrity and consistency axioms.
It is now possible to develop the concept of a multi-user relational database system.

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

**Definition:** A multi-user relational database schema is a triple \( \mathcal{M} = (L, B, U) \), where \( L = (S, D, F, R, T, A, L) \) is a relational database schema with semantics, \( B \) is a partial interpretation schema which is dense, and \( U = (U_1, \ldots, U_n) \) is a set of partial interpretation schemata such that \( \{B, U_1, \ldots, U_n\} \) is a partition of \( R \). \( U \) is called the set of views of \( M \). \( B \) is the base view schema; the \( U_i \)'s are called the user view schemata. A state of \( M \) is a just a model of \( L \). Given a state \( I \), the restriction of \( I \) to \( B \) (resp. \( U_i \)) is called the base state or \( B \)-state (resp. 4th user state or \( U_i \)-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, \( B = (R_1, R_2, R_3), U = (U_1, U_2) \), where \( U_1 = (R_4), U_2 = (R_5) \).

3. Information, Data, and the Update Strategy
For the rest of this paper, fix a multi-user relational database schema \( \mathcal{M} = (L, B, U) \) with \( L = (S, D, F, R, T, A, L) \).

**Definition:** Let \( \mathcal{Q} \) be a view of \( \mathcal{M} \), and let \( I \) be a state of \( \mathcal{Q} \).
(a) The information of the corresponding \( \mathcal{Q} \)-state, denoted \( \text{Info}(I, \mathcal{Q}) \), is the set of all sentences in \( \text{I-span}(\mathcal{Q}) \) which are satisfied in \( I \).
(b) The data of the corresponding \( \mathcal{Q} \)-state, denoted \( \text{Data}(I, \mathcal{Q}) \), is the set of all relations \( R^k \) with \( R \subseteq \mathcal{Q} \).

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

It is important to know when \( \mathcal{B} = \text{Info}(I, \mathcal{Q}) \) for a given model \( I \). This is easily answered, as \( \mathcal{B} \) is the set of \( \text{Info}(I, \mathcal{Q}) \) if and only if \( \mathcal{B} \) is data-determining, since \( R(c_1, \ldots, c_n) \in \text{Info}(I, \mathcal{Q}) \) if and only if \( (c_1, \ldots, c_n) \in R \).

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

User View of Data -------- Base View of Data  
Information in User View  Information in Entire System  
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, \( I \) and \( J \). 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 \( J \).

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

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

Step 3b: Check to see if the update \( I \) will satisfy the integrity constraints. If not, the update cannot be made.

Step 3c: Determine any secondary (user view) information updates from \( I \); insert this information
Step 3d (Path 3): Perform the data update corresponding to IUJ.

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 ReUI for some i, and T, T1, and T2 are tuples of the appropriate rank. It shall also be assumed that each specified update is non-trivial, 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 $\mathcal{M}$, all axioms in $\mathcal{A}_{\mathcal{M}}$ shall be in D-span($\mathcal{E}$).

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.

\[
\begin{align*}
&\forall x \forall y (\exists z ((R(x,z) \land T(x,y)) \lor (\exists z ((R(x,z) \land T(x,y))))),
&\forall x \forall y (\exists z ((R(x,z) \land T(x,y)) \land \exists z ((R(x,z) \land T(x,y)))).
\end{align*}
\]

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,\text{Gamma}) into R5;
(4) Delete (Best,101,\text{Alpha}) from R5;
(5) Replace (Acme,101,\text{Alpha}) with (Acme,101,\text{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 $\mathcal{J}$ are as follows.

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

$\mathcal{J}$ is given below for each of the five examples.

(1) $\mathcal{J} = \{\neg R4(101,10)\}.
(2) \mathcal{J} = \{R4(181,3)\}.
(3) \mathcal{J} = \{R5(A1,105,\text{Gamma})\}.
(4) \mathcal{J} = \{\neg R5(\text{Best},101,\text{Beta})\}.
(5) \mathcal{J} = \{\neg R5(\text{Acme},101,\text{Alpha}), R5(\text{Acme},101,\text{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 $\mathcal{J}$ 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 $\mathcal{J}$ is bound to the consistency constraint in which it occurs. Variables
in the context are bound to the corresponding constants in the elemental sentence. The result is a primar 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 $(\forall x)(\forall y)(R4(x,y) \land R1(x,z,y))$. First, $R4(101,110)$ is bound to this constraint to get $(\exists z)R1(101,110,z)$. 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 $I = (\neg \forall z)R4(101,10)z$.

The other four examples proceed in an essentially similar fashion. The results are given below.

(2) $I = (\exists z)R1(181,1,z,3)$.
(3) $I = R2(A1,105) R3(105,\text{Gamma})$.
(4) $I = (\neg R2(\text{Best},101) \land R3(101,\text{Beta}))$.
(5) $I = (\neg R2(\text{Acme},101) \land R3(101,\text{Alpha}))$.

Expansion of Step 3a
The idea of the information update is to take the sentences in $I$ and update the information in $\text{Info}(I,B)$ with $I$. Now $\text{Info}(I,B)$ is data determining, 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 $\text{Info}(I,B)$ 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 cannot 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 $\neg (\exists z) R1(101,110,x)$, which is the same as $(\forall x)(\neg R1(101,110,x))$. This has the elemental basis $(\neg R1(101,\text{Widget},10))$, $R1(101,\text{Franzoi},10)$, $R1(101,\text{Thing},10)$, $R1(101,\text{X-J},10)$, $\neg R1(101,\text{KP-7},10)$. Note that the conversion required to use all of elements from the PARTNAME domain, and that the fact that this domain is finite is crucial.

(2) The only sentence in $I$ is $(\exists z) R1(181,1,z,3)$. It has no elemental basis. 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 $R2(A1,105)$ $R3(105,\text{Gamma})$, which clearly has the elemental basis $(R2(A1,105), R3(105,\text{Gamma}))$.

(4) The only sentence in $I$ is $(\neg R2(\text{Best},101) \land R3(101,\text{Beta}))$. It has no elemental basis.

(5) The sentences in $I$ are $(\neg (R2(\text{Acme},101) \land R3(101,\text{Alpha}))$ and $R2(\text{Acme},101) \land R2(101,\text{Epsilon})$. The first is equivalent to $(\neg R2(\text{Acme},101))$. However, the second implies that $R2(\text{Acme},101)$ is true, so $(\neg R3(101,\text{Alpha})$ must be true. Hence I has the elemental base $(\neg R3(101,\text{Alpha}), R2(\text{Acme},101), R3(101,\text{Epsilon}))$.

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 dependences 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 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 integrity constraints are bound to primary clauses. The examples which have survived this far are now presented.

(1) Using the constraint $(\forall x)(\forall y)(R4(x,y) \land R1(x,z,y))$, the elements of I are in turn bound to the right-hand side to give $I = (\neg R4(101,10))$, exactly as the previous I.

(2) Using the constraint $(\forall x)(\forall y)(R5(x,y,z) \land R2(x,y) \land R3(x,y,z))$, binding I to the right-hand side yields $(R5(A1,105,\text{Gamma}), R5(\text{Acme},105,\text{Gamma}))$. Note that the binding required reference to $\text{Info}(I,B)$, to determine the satisfiability of the primary clause.

(3) Using the same constraint as in (3), binding I to the right-hand side yields $I = (\neg R5(\text{Acme},105,\text{Gamma}), R5(\text{Best},101,\text{Alpha}), R5(\text{Acme},105,\text{Epsilon}))$.

Expansion of Step 3d
Using the correspondence between the elemental sentences and data, this step is trivial. The examples are as follows:

(1) Delete $(101,\text{Widget},10)$ from Rel. 1 and delete $(101,10)$ from Rel. 2.

(2) Insert $(A1,105)$ into Rel. 2, $(105,\text{Gamma})$ into Rel. 3, and $(A1,105,\text{Gamma})$ and $(\text{Acme},105,\text{Gamma})$ into Rel. 2.

(3) Delete $(101,\text{Alpha})$ from Rel. 3; insert $(101,\text{Epsilon})$ into Rel. 3; delete $(\text{Acme},101,\text{Alpha})$ and $(\text{Best},101,\text{Alpha})$ from $R5$; insert $(\text{Acme},101,\text{Epsilon})$ and $(\text{Best},101,\text{Epsilon})$ 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 (x,y)R1(181,x,y). 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


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.