Updates and Set Identities

David McGoveran

In 1994, Date and McGoveran (Ref. 3) introduced an approach to update[1] of relational expressions (the Date-McGoveran update rules hereinafter) that has been the subject of considerable skepticism and debate. The approach has been refined over the intervening years, but retains its essential features. The essence of the approach is to exploit the set theoretic properties of relations, including their membership functions, to determine how a collection of relations can be updated consistently. Necessarily, the update procedure must apply to any relational expression, even if that expression involves multiple relations.

It follows from the fact that relations are sets that the relational expression can be either side of a set identity, and the evaluation of the updated RHS (right hand side) and LHS (left hand side) must be equivalent. Two problems seem to arise in implementing these conclusions:

1. In a number of set identities (reduction formulas) the numbers of sets on one side of the identity does not match that on the other side. Thus, it would appear that using set identities to simplify a relational expression could have a side effect of eliminating the update of one or more relations that would have been updated in the original expression.

2. There seems to be something wrong with the notion that relations are sets. After all, set theory does not have update operations. Furthermore, in set theory, if two sets have the same membership function, they are indistinguishable, whereas this may not be the case in the common understanding of relational theory and practice.

To understand, and resolve, these problems we will take a specific example of a set identity (Equation 1:

$$\mathbf{A \cup (A \cap B) \equiv A}$$

The LHS of this identity references the set B, while the RHS references only A. According to the Date-McGoveran update rules, if these sets are relations, then an update of relational expression involving symmetric set operations (like union and intersection) on A and B must be applied equally to A and B. That is, the update algorithm must not be sensitive to the order of A and B in the expression. On the surface, this seems to imply

[1] We use the term update to refer to any of set deletion, set insertion, or set modification, traditionally presented as the operations DELETE, INSERT, and UPDATE. Note however that we understand these quite differently from the SQL operations of the same names which are neither as consistent or powerful as they could be, and considerably more complex and difficult to use than they need be.

that the Date-McGoveran update rules will update B if the LHS expression is used, but only A if the RHS expression is used, violating the identity.

To see that this is not the case, we need to understand the possible relationships between sets (relations) A and B vis-à-vis their membership functions (i.e., relvar predicates). Recall from earlier papers that an update is always characterized by a predicate and a relation by its relvar predicate. Without going into detail, recall also that both an update predicate and a relvar predicate can reference the current value of the relvar and that the relation must always satisfy its relvar predicate (i.e., both before and after the update). Note that, in understanding a set relationship among relations, we focus the relvar predicates and insist on set updating rules. Update algorithms proposed before those of Date-McGoveran focused on the relvar value and often on anticipating a modification of the values of one or more specific rows. The Date-McGoveran update rules offer no concept of modifying, inserting, or deleting a row, since they enforce the semantics of set transformations. If a set transformation affects any particular row, its does so solely because the update predicate applies to that row.

Because we are concerned with relations, we may assume that a meaningful binary set operation means the relations are compatible for our purposes. We will focus on intersection, which means – according to the Date-McGoveran update rules – that the update predicate must satisfy the logical conjunction of RA (the relvar predicate for A) and RB (the relvar predicate for B). There are three possibilities dictated by the relation membership functions, each of which we address in turn:

1. $A \cap B \equiv 0$ (A necessarily disjoint with B)

2. $A \cap B \equiv A$ and $A \cap B \equiv B$ (hence A necessarily identical to B)

3. $A \cap B \neq 0$ (A and B necessarily non-disjoint)

Case 1: If A and B are disjoint, then their relvar predicates are disjoint so that any tuple may satisfy RA or RB but not both. (Note that we do not mean that the relation values are disjoint, as this is irrelevant to the update rules.) In this case, although an update predicate may be applied to the intersection of A and B, it cannot have any effect since it cannot satisfy $R(A \cap B) \equiv$ RA AND RB.[2]

Case 2: If A and B are necessarily identical, then they have identical relvar predicates. As will be discussed below, this means that B does not exist except as a renaming of A (or vice-versa).

Case 3: The most interesting case is that A and B are necessarily non-disjoint. That is to say, we may understand A as comprised of subsets A1 and A2, and B as comprised of subsets B1 and B2. Subset A1 is that portion of A that is necessarily disjoint with B and subset B1 is that portion of B that is necessarily disjoint with A. Thus, these subsets

[2] Recall that the Principle of Orthogonal Design (Ref. 2) requires that any two base relvars in a database satisfy this condition. Thus, in a database so designed, only Case 1 need be considered.

follow Case 1 above. Subset A2 is necessarily identical to subset B2 by the definitions, so the update of these portions follows Case 2 above. Thus, Case 3 reduces to a combination of Cases 1 and 2.

Now let’s return to the second problem mentioned above. While Equation 1 applies to any sets A and B of simple set theory, it clearly does not apply to any relations A and B of relational theory as commonly understood. In particular, relational union and intersection operations require some notion of relation or relvar type compatibility to be meaningful and implementable. The traditional notion of “union-compatibility” is such a rule, but one we find overly simplistic, overly informal, and insufficiently precise. Instead, relvar predicates, their logical relationships, and logic operations among them form the basis of our relvar type system and control whether or not relations can be combined by set operations.

One might object that this is a departure from simple set theory semantics with its seemingly unconstrained operations permitted among sets. However, this presumed property of simple set theory is an illusion. Part of what makes simple set theory “simple” is the assumption of a universe of discourse U. All sets (such as A and B) are subsets of U, and thus are implicitly type compatible. It is this implicit type compatibility that makes the familiar Venn diagram formalism valid. Creating or designating a subset A of U implicitly asserts the creation of membership function that is more restrictive than the membership function of the set. Thus, the subset A is a subtype of the type of U. The set identities of simple set theory do not type sets, but assume for purposes of the formula that all sets have the same type as U (i.e., their supertype).

The importance of this is that, when applying a simple set identity, we must apply simple set theory semantics to our understanding of the identity. The implications include:

- Set elements (tuples in relational theory) are distinguishable and countable only if their identified properties differ, and a element labels are not element properties.

- Sets (relations in relational theory) are distinguishable and countable only if their identified properties differ, and a set labels are not element properties.

- If two sets are defined so as to contain the same members, then they are identical (not just equivalent by membership) and there is in fact exactly one set.

These aspects of the formalism of simple set theory imply that, in Case 2 above, A and B should be replicas of each other. Hence the only way they can have different relvar values would be that they were updated by name rather than by relvar predicate. This is a violation of the update rules because it means that name is substituting for a presumed difference between A and B that is not captured by the relvar predicates. (A discussion of ways to create a pair of relvar predicates that capture a tacit difference between two relations is beyond the scope of this paper. For now, assume that it can be done so there is never a need to rely on relvar or relation names.) Asuming the rules are followed, the only way this case can occur is if A and B are redundant relations In our version of relational semantics, names are merely convenient shorthands for relvar predicates and so the appearance of such redundant A and B is an illusion.

In summary, even if A and B differ only by name in the database so that an update applied to the LHS of Equation 1 results in, for example, the insert of a set of rows to B, we are guaranteed that any such rows provide no more information than if that set of rows had been inserted into the RHS alone. They are completely redundant. Similar conclusions follow for set deletions and set modifications.

The above shows once again why using names as the only differentiator of two relations is worse than a bad idea, it is a violation of set semantics. Not only does it lead to confusion for users of a database who cannot intuit how A and B differ (except by the fact of relvar value), it also hides critical definitional information from the DBMS and makes semantic optimizations possible only through a patchwork of code to handle special cases. We believe strongly that, if it is necessary to have A and B as distinct in a database, then it is equally necessary to capture that difference as a difference in the relvar predicates. Doing anything less can only be due to laziness, ignorance (i.e., inadequate relational database design skills), or obstinance.

1. Date, C. J., and McGoveran, D., "Updating Joins and Other Views," DataBase Programming and Design, August, 1994. Also in C. J. Date, Relational Database Writings 1991-1994, Addison-Wesley, ©1995.

2. Date, C. J., and McGoveran, D., "A New Database Design Principle," DataBase Programming and Design, July, 1994. Also in C. J. Date, Relational Database Writings 1991-1994, Addison-Wesley, ©1995.

3. Date, C. J., and McGoveran, D., "Updating Union, Intersection, and Difference Views," DataBase Programming and Design, June, 1994. Also in C. J. Date, Relational Database Writings 1991-1994, Addison-Wesley, ©1995.