THE RELATIONAL ALGEBRAIC OPERATOR NOT

by

Yo'av Raz

Technical Report #421
September 1986
THE RELATIONAL ALGEBRAIC OPERATOR NOT

by

Yoav Raz
Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT

Some Relational Algebra (RA) operations give the exact meaning of Natural Language (NL) constructs when both Entity-sets and Relationship-sets are represented as relations (e.g. relativization and the natural-join, coordination and the union and intersection, universal-quantifiers and set-comparison, and the generalized-division, restriction and the selection and theta-join). The proposed NOT (COMPLEMENT) operator enables to express NL negation as a single operation in "closed world" databases. It can replace the subtraction, which does not have a direct linguistic analog, and is useful being implemented in an environment which supports NL or NL-like queries using RA.

Keywords: Relational Algebra, Relational Databases, Query Languages, Natural Language Queries.
1. INTRODUCTION

Some Relational Algebra (RA) operations give the exact meaning of Natural Language (NL) constructs when both Entity-sets and Relationship-sets are represented as relations (e.g. relativization and the natural-join, coordination and the union and intersection, universal-quantifiers and set-comparison and the generalized-division, restriction and the selection and theta-join - see [MR1], [MR2], [Raz]; see also [Win] for detailed constructs' specifications). The proposed NOT (COMPLEMENT) operator enables to express NL negation as a single operation in "closed world" databases ([Rei]). It can replace the subtraction, which does not have a direct linguistic analog, and is worthwhile being implemented in an environment which supports NL or NL-like queries using RA.

In what follows the NOT operator is defined and its properties are demonstrated. Related algebraic identities are demonstrated graphically rather than being proved formally.

In Part 2 the NOT operator is defined and examined. Part 3 discusses the linguistic aspect of a relationship (connection) relation. In Part 4 the analogy between the NOT and negation in Natural Language is explained and demonstrated. Part 5 shows the form De Morgan laws take for the NOT, UNION and INTERSECTION. The paper concludes in Part 6 where it is shown that the NOT operator can replace SUBTRACTION in a set of RA operators which includes also UNION (or INTERSECTION) and PROJECTION.
2. THE NOT OPERATOR

The NOT (COMPLEMENT) operator produces all the tuples in the Cartesian product of a given set of unary relations (considered parameters) over attribute names of the operand, which do not appear in the projection of the operand over the attribute names of the above relations.

Definition:
Let $A$ be a set of attribute names, and $R$ a relation over $A$: $R = R(A)$;
also $R_x = R_x \{x\}$, $x \in A' \subseteq A$.

Then the complement (negation) of $R$ relatively to $\{R_x \mid x \in A'\}$ is the following:

$$\text{not}_{\{R_x \mid x \in A'\}}(R) = \times_{x \in A'} R_x - R[A']$$

where "\times" stands for the regular Cartesian Product, ",-" stands for the regular Subtraction and $R[A']$ is the Projection of $R$ over $A'$ (See [Pir], [Üll] for definitions).

A graphical interpretation of the NOT operator is given in Fig. 1.

\[
\begin{align*}
R &= R(x, y, ...) \\
R_x &= R_x(x) \\
R_y &= R_y(y)
\end{align*}
\]

![Diagram of NOT operator]

Fig. 1
Notations:

1. In what follows the notation $\textit{not}$ stands for $\textit{not} \{R \mid x \in A'\}$.

2. When attribute names are specified the notation $R(x,y,\ldots)$ is used instead of $R((x,y,\ldots))$.

Note that the $\textit{NOT}$ operator implies a projection over $A'$. Hence, it commutes with projection, and an explicit projection over any $A'' \supset A'$ can be eliminated:

Lemma 1:
Let $A \supset A'' \supset A'$ then

$$\textit{not}(R) = \textit{not}(R[A'']) = (\textit{not}(R))[A''].$$

Applying twice the same $\textit{NOT}$ operator to a relation $R$ does not return $R$ in the general case:

Lemma 2:

$$\textit{not}(\textit{not}(R)) = (\times_{x \in A'} R_x) \cap R[A']$$

However, of course,

Corollary 1:

If $\times_{x \in A'} R_x \supset R[A']$ then

$$\textit{not}(\textit{not}(R)) = R[A']$$

and $R$ is returned if and only if $A' = A$.

3. THE LINGUISTIC ASPECT OF THE INFORMATION IN THE DATABASE

It is common among DB designers to describe databases on its conceptual level using Entity (Object) - Relationship (association) Diagrams (ERDs) ([Che]) which are a kind of semantic networks ([BF]) used for knowledge representation. Informally, an ERD is a graph with three types of nodes:
1) Entity sets. 2) Relationship sets. 3) Attributes.

The above sets consist of elements of the same type (with the same attributes). The arcs connect entity sets with appropriate attributes and relationship sets, and also relationship sets with their attributes. An example
of an ERD appears in Fig. 2. The notion of ERD can be extended beyond what demonstrated here to capture more of the real world's semantics. However, even with its simple version the basic ideas can be presented.

A relationship between entities can usually be described by a simple sentence with an entity as the subject part, the other entities participating in the relationship as objects and a predicate part which usually induces the relationships' name.

For example, a member in the relationship set SUPPLY can be described by the sentence

SUPPLIER SUPPLIES ITEM to DEPARTMENT. or

DEPARTMENT IS SUPPLIED by SUPPLIER with ITEM.

If the relationship has attributes, they may be combined in the sentence:

SUPPLIER STOCKS QUANTITY of ITEM.

The association between an entity and its attribute may be expressed as

SUPPLIER HAS NAME... or

NAME OF SUPPLIER.

Some of the RA operators can take advantage of the ERD's linguistic aspect by calculating relations which contain tuples satisfying predicates expressed in natural language sentences based on the sentences derived from the ERD. As we shall see the proposed NOT operator is the natural one to take care of negation.
DEPARTMENT = DEPARTMENT(No, NAME, FLOOR);  key=No.
ITEM = ITEM(No, NAME, COLOR, TYPE);  key=No.
SUPPLIER = SUPPLIER(No, NAME, LOCALITY);  key=No.
REQUEST = REQUEST(DEPARTMENT-key, ITEM-key, QUANTITY)
STOCK = STOCK(ITEM-key, SUPPLIER-key, QUANTITY)
SUPPLY = SUPPLY(ITEM-key, SUPPLIER-key, DEPARTMENT-key, QUANTITY, PRICE)

The relations representing the ERD in Fig. 2.
4. NEGATION AND THE NOT (COMPLEMENT) OPERATION

The analogy between Natural Language negation and the NOT operation (and also between other NL constructs and respective operations) is exposed when both entity sets and relationship sets are expressed as relations in the most straightforward way (see Fig. 3):

For each entity set there is a relation whose attributes are these of the entities in the set;
For each relationship there is a relation whose attributes are these of the relationships in the set, augmented with the key attributes in all the entity sets associated by the relationship (key attributes are such that given its values, an entity is uniquely specified. Since a key in a relationship is used as an identifier only, we shall view all the key attributes of an entity appearing in a relationship as a single attribute with values consisting of concatenating key attribute values of entities in a certain defined order).

When the word "NOT" is added to a sentence appropriately the meaning of the sentence turns to its logical complement. Assuming that the DB includes true facts and what is not included is not true (The closed world assumption [Rei]), the complement operator enables us to compute the opposite meaning of a sentence.

As we have seen, the meaning of

SUPPLIER SUPPLIES ITEM TO DEPARTMENT.

is given by the relation SUPPLY (See also [Bil]). In SUPPLY all the facts about "who" supplies "what" to "whom" are kept. The meaning of the opposite sentence

The SUPPLIER DOES NOT SUPPLY ITEM TO DEPARTMENT.

is all the relevant triples of supplier, item and department which do not appear in SUPPLY. These tuples are derived by the following expression:

\[ \text{not} \{ \text{SUPPLIER[SUPPLIER-key], ITEM[ITEM-key], DEPARTMENT[DEPARTMENT-key]} \} (\text{SUPPLY}) \].

If we are interested in the list of relevant suppliers only, a projection by SUPPLIER-key should be added. Note that since SUPPLY connects entities already in the database, consistency constraints imply that

SUPPLIER[SUPPLIER-key] ⊇ SUPPLY[SUPPLIER-key],

and the same for the keys of ITEM and DEPARTMENT.

As another example let us examine the following restriction:

A RED ITEM.
The meaning of such statement is given by selecting from the relation ITEM all items with COLOR="RED", i.e.

\[(\text{select}_{\text{COLOR}=\text{RED}}(\text{ITEM}))[\text{ITEM}-\text{key}]\].

The meaning of its negation,

NOT A RED ITEM

is achieved by applying the not operator:

\[\text{not}_{\text{ITEM}}(\text{ITEM}-\text{key})(\text{select}_{\text{COLOR}=\text{RED}}(\text{ITEM}))\]

which is identical to

\[(\text{select}_{\text{COLOR}=\text{RED}}(\text{ITEM}))[\text{ITEM}-\text{key}]\]

representing the equivalent sentence

AN ITEM WHICH IS NOT RED.
5. DE-MORGAN LAWS FOR THE NOT OPERATOR

The De-Morgan laws for the NOT operator are the following:

Lemma 3:
Let $R_1 = R_1(A)$, $R_2 = R_2(A)$.

Then

$$\neg(R_1 \cup R_2) = \neg(R_1) \cap \neg(R_2)$$

and

$$\neg(R_1 \cap R_2) = \neg(R_1) \cup \neg(R_2)$$

The first law is demonstrated in Fig. 4.

\[ R_1 = R_1(x, y) \]
\[ R_2 = R_2(x, y) \]
\[ R_x = R_x(x) \]
\[ R_y = R_y(y) \]

Fig. 4
As a result of Corollary 1 and Lemma 3 the following two sentences are equivalent under the above semantics:

ITEM WHICH IS NOT RED AND WHICH IS SUPPLIED BY SUPPLIER TO A DEPARTMENT.

NOT (ITEM WHICH IS RED OR WHICH IS NOT SUPPLIED BY SUPPLIER TO A DEPARTMENT).

The meaning of them is given by the following two identical expressions respectively:

\[(\neg ITEM \land \neg \text{select}_{\text{COLOR}=\text{RED}}(\text{ITEM})) \land \neg \text{SUPPLY} \]

and

\[\neg ITEM \land \neg \text{select}_{\text{COLOR}=\text{RED}}(\text{ITEM}) \land \neg \text{SUPPLY}\]
6. EXPRESSING SUBTRACTION USING THE NOT OPERATOR

The relational subtraction operation can be expressed using the NOT operator:

Lemma 4:

Let \( R_1 = R_1(A), \ R_2 = R_2(A) \) be two (union-compatible) relations. Then

\[
R_1 - R_2 = \text{not}_{[\{x\} | x \in A]}(R_1 \cap R_2) \cap R_1 =
\]

\[
= \text{not}_{[\{x\} | x \in A]}\left(\text{not}(R_1) \cup \text{not}(R_2) \right) \cup \text{not}(R_1)
\]

Lemma 4 is demonstrated in Fig. 5.

Lemma 4 implies that the NOT operator can replace Subtraction in a set of operators which includes Union (or Intersection) and Projection.

\[
R_1 = R_1(x, y) \\
R_2 = R_2(x, y)
\]

Fig. 5
REFERENCES


