Some Operations on Database Universes

E.O. de Brock

September 1997

SOM theme A: Intra-firm coordination and change

Abstract

Operations such as integration or modularization of databases can be considered as operations on database universes. This paper describes some operations on database universes. Formally, a database universe is a special kind of table. It turns out that various operations on tables constitute interesting operations on database universes as well.

Keywords: Databases, database universes, operations on universes.

E.O. de Brock
Faculty of Management and Organization
State University of Groningen
P.O. Box 800, 9700 AV Groningen
The Netherlands
Tel. +31.50.3637315
Fax +31.50.3632275
Email: e.o.de.brock@bdk.rug.nl

Introduction

In practice, we are often concerned with integration of databases, modularization of databases, renaming tables, and so on. These manipulations can be considered as operations on database universes. Therefore, we are interested in operations that assign to one or more DB universes a DB universe again.

Since a DB universe is a set, notions such as subset, union, intersection and difference also apply to DB universes. For the most part, such operations yield DB universes once again, namely over the original DB schema; the reader is referred to the cases (1), (2), and (3) of Theorem 2 below for details.

Furthermore, Theorem 1 states that each DB universe over a set-valued function g is also a table over dom(g). This means, therefore, that all concepts, notations and operations that have been defined for tables, apply to DB universes as well!
Hence, in particular, the table operations of projection, natural join, and renaming are also applicable to DB universes, each producing tables, according to well-known results. But does the application of such an operation to DB universes yield a *DB universe* once again, and, if so, over which DB schema? From part (4) of Theorem 2 on, we will investigate this matter, resulting in several interesting results.
First of all, however, we will briefly describe what those operations on tables amount to, when applied to DB universes:

- Projection (on a set of table names) corresponds to the "restriction" of the DB universe (and of the DB states) to a "module" or "subscheme" of table names (that are related in some sense).
- The join of two DB universes having joinable DB schemas corresponds to the "integration" of all possible pairs of DB states having the same (table) value for the same table names. Each "integration" results in a DB state containing all table names of the two original DB states.
- Finally, renaming in the case of DB universes amounts to the renaming of tables.

1. Basic notions

In this section, we start by establishing our definitions of the important basic notions of table, database state, and database universe.

By a table over a set A we mean a set of functions over A, i.e., functions with A as their domain:

Definition 1:
If A is a set, then:
T is a **table over** A $\Leftrightarrow$ T is a set and $\forall$ t $\in$ T: t is a function over A.

An element of T is called a *tuple* and an element of A is called an *attribute* of T.

Since every table is a (special) set, every concept defined for sets also applies to tables. Thus, for example, the notions of union and intersection of two tables make sense.

A database skeleton (or database schema) can be considered as a set-valued function, assigning to each table name its set of attributes. As a frame of reference we present the simple but also well-known and widely used example in the database literature concerning suppliers, parts and shipments (cf. [Da 95]). The suppliers/parts/shipments-example has the following DB skeleton, which we will call g1:

```
g1 = {  (S  ; {S#, SNAME, STATUS, CITY}),              | suppliers
        (P  ; {P#, PNAME, COLOR, WEIGHT, CITY}),       | parts
        (SP ; {S#, P#, QTY})}                          | shipments
```

Here S# stands for supplier number, P# for part number, and QTY for quantity.

We define the concept of a DB state for any set-valued function g:

Definition 2:
If g is a set-valued function, then:
v is a **DB state over** g ⇔ v is a function over dom(g) and
$\forall$ E ∈ dom(g): v(E) is a table over g(E).

Since every DB state is a function, every concept defined for functions also applies to DB states. Thus we can speak about the domain and the range of a DB state, for instance. In our example above, dom(v) = {S, P, SP} for every DB state v over g1.

The set of admissible states (to be determined by the organization in question) is therefore some set of DB states over g1. We call such a set a database universe (or briefly DB universe) over g1. In general:

Definition 3:
If g is a set-valued function, then:
U is a **DB universe over** g ⇔ U is a set of DB states over g.

If U is a DB universe over g, then we call g *the DB skeleton* (or "database schema") *of* U, an element E of dom(g) a *table index* (or "table name") *of* U, g(E) *the heading of* E *in* U, and an element of g(E) an *attribute* (or "attribute name") *of* E *in* U. We call an element of U a *DB state consistent with* U.

Since every DB universe is a (special) set, each concept defined for sets applies to DB universes as well. Moreover, formally speaking, a DB universe over a set-valued function g is also a table over dom(g):

Theorem 1:
If g is a set-valued function and U is a DB universe over g, then U is a table over dom(g).

Proof:
From Definition 3 it follows that U is a set, and
from Definition 2 it follows that each element of U is a function over dom(g).
Hence U satisfies Definition 1.

Thus, the table indices of U act as the attributes of the "super table" U, the DB states as the tuples of that table, and the tables in such a DB state as the (non-1NF) "attribute values" in the table U. For example, any DB universe over g1 is a table over {S, P, SP}.

We will denote *function composition* by $f \circ g$ (f *after* g). We call the functions f and g *joinable* iff f and g match on $dom(f) \cap dom(g)$, i.e., iff $\forall\ a \in dom(f) \cap dom(g) : f(a) = g(a)$.

The *restriction* of a tuple t to an attribute set B is denoted by $t \lceil B$ and the remaining elements constitute the set $t \lfloor B$:
$t \lceil B = \{ (a;w) \mid (a;w) \in t \text{ and } a \in B \}$
$t \lfloor B = \{ (a;w) \mid (a;w) \in t \text{ and } a \notin B \}$

The *projection* of a table T on an attribute set B is denoted by $T \Vert B$ and the remaining columns constitute the table $T \Downarrow B$:
$T \Vert B = \{ t \lceil B \mid t \in T \}$
$T \Downarrow B = \{ t \lfloor B \mid t \in T \}$

The *natural join* of the tables T and T' is denoted by $T \bowtie T'$:
$T \bowtie T' = \{t \cup t' \mid t \in T \text{ and } t \in T' \text{ and } t \text{ and } t' \text{ are joinable } \}$.

Another useful way to obtain a table from another table is by substituting new "column names" for "old" column names. We are now faced with the question of how to formally define this operation on tables. For that purpose we use an *attribute transformation* (or "renaming function") h that assigns to each "new" column name b the old column name that has to be replaced by b. For instance, suppose that we want to obtain a table over {SUPNO, PARTNO, QTY} by renaming the attributes S# and P# of the SP-table of a given DB state v over the DB skeleton g1 described before. Then we can use the following attribute transformation h1 in combination with the table v(SP):

h1 = {(SUPNO;S#), (PARTNO;P#), (QTY;QTY)}

We denote the "renamed table" in this case as $v(SP) \infty h1$. In general, the formal definition is as follows:

If T is a set of functions and h is a function, then:
$T \infty h = \{ t \circ h \mid t \in T \}$

As an illustration we give a representation of a table T1 over {S#, P#, QTY} and the resulting table T1 ∞ h1 over {SUPNO, PARTNO, QTY}, where we applied the attribute transformation h1 above:

T1:

S#	P#	QTY
S1	753	100
S1	901	388
S2	467	467

T1 ∞ h1

SUPNO	PARTNO	QTY
S1	753	100
S1	901	388
S2	467	467

2. Operations on Database Universes

The following theorem contains the main results of this paper.

Theorem 2:
If g and g' are set-valued functions, U and V are DB universes over g, U' is a DB universe over g', $D \subseteq U$ and $X \subseteq dom(g)$, and h is a function, then:
(1) D is a DB universe over g;
(2) $U \cap V$ is a DB universe over g and
$U - V$ is a DB universe over g;
(3) $U \cup V$ is a DB universe over g;
(4) $U \Vert X$ is a DB universe over $g \lceil X$;
(5) $U \bowtie U'$ is a DB universe over $g \cup g'$ provided that g and g' are joinable;
(6) $U \infty h$ is a DB universe over $g \circ h$;
(7) $U \Downarrow X$ is a DB universe over $g \upharpoonright X$.

Proof:

(1) Each element of U is a DB state over g, so each element of D is one too.

(2) This follows immediately from (1).

(3) Each element of U is a DB state over g and
each element of V is a DB state over g,
so each element of $U \cup V$ is one as well.

(4) Each element of $U \restriction X$ is of the form $v \restriction X$, for some $v \in U$.
It holds that $dom(v \restriction X) = X = dom(g \restriction X)$,
hence $v \restriction X$ is a function over $dom(g \restriction X)$.
It also holds for each $E \in dom(g \restriction X)$ that $(v \restriction X)(E) = v(E)$.
Now v(E) is a table over g(E),
so $(v \restriction X)(E)$ is a table over g(E), i.e., over $(g \restriction X)(E)$.
Then $v \restriction X$ is a DB state over $g \restriction X$, according to Definition 2.
Then $U \restriction X$ is a DB universe over $g \restriction X$, according to Definition 3.

(5) Each element of $U \bowtie U'$ is of the form $v \cup v'$, where $v \in U$ and $v' \in U'$ and
v and v' are joinable. It holds that
$dom(v \cup v') = dom(v) \cup dom(v') = dom(g) \cup dom(g') = dom(g \cup g')$,
hence $v \cup v'$ is a function over $dom(g \cup g')$.
It also holds for each $E \in dom(g \cup g')$ that
- $E \in dom(v)$, hence $(v \cup v')(E)=v(E)$, being a table over g(E), or
- $E \in dom(v')$, hence $(v \cup v')(E) = v'(E)$, being a table over g'(E).

Since g and g' are joinable,
in either case $(v \cup v')(E)$ is a table over $(g \cup g')(E)$.
Then $v \cup v'$ is a DB state over $g \cup g'$, according to Definition 2.
Then $U \bowtie U'$ is a DB universe over $g \cup g'$, according to Definition 3.

(6) Each element of $U \infty h$ is of the form $v \circ h$, where $v \in U$.
Since dom(v) = dom(g), we have that $dom(v \circ h)= dom(g \circ h)$,
hence $v \circ h$ is a function over $dom(g \circ h)$.
We also have for each $E \in dom(g \circ h)$ that $(v \circ h)(E) = v(h(E))$,
which is a table over g(h(E)), i.e., over $(g \circ h)(E)$.
Then by Definition 2, $v \circ h$ is a DB state over $g \circ h$.
Then by Definition 3, $U \infty h$ is a DB universe over $g \circ h$.

(7) The proof of this part is analogous to the proof of (4).

In the literature we find various properties of the operations treated in Theorem 2, applied to tables (e.g., [Da 95] or section 2.1.1 of [Br 95], which contains extensive lists of properties). Note that these properties remain valid when applied to DB universes!

Conclusion

When looking for operations on database universes, it turns out that various operations on tables constitute interesting operations on database universes as well. Operations such as integration or modularization of databases are examples of such operations on database universes.

References

[Br 95] E.O. de Brock: *Foundations of semantic databases.*
Prentice Hall International Series in Computer Science, London, 1995

[Da 95] C.J. Date: *An introduction to database systems.*
Addison-Wesley, Reading (Mass.), 1995