Universal Relation Views: A Pragmatic Approach

Joachim Biskup and Hans Hermann Brüggemann

Informatik III
University of Dortmund, West Germany

Abstract:

A universal relation view is a view (external schema) on top of a relational database schema (conceptual schema). It shows the whole database as a single fictitious universal relation. It provides a user-friendly interface where queries can be expressed without navigation, even on the level of relations, as far as possible. We carefully develop the fundamental decisions of our approach: different kinds of users should have a consistent understanding of the view; the underlying database schema should satisfy a new desirable property called unambiguity. Besides unambiguity and the so-called universal scheme assumption our approach does not rely on any further prerequisites.

1. Introduction

A user can access to information of a database by means of a data manipulation language. In case of a tuple-oriented (one record at a time) language he must tuple-wise "navigate" in the database, for instance by using the FIND-statement of DBTG network databases. In case of a high-level, relation-oriented language, as for instance the relational algebra for relational databases, the burden of tuple-wise navigation is taken by the database system. However, the user must still relation-wise navigate, in particular he must determine appropriate join paths in the database schema.

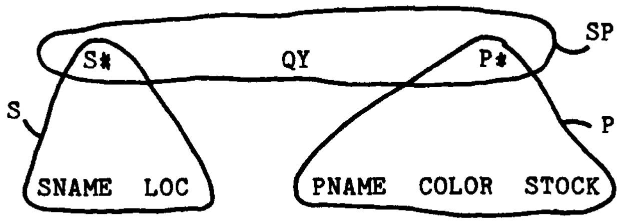
As an example, consider (a variation of) the well-known suppliers-parts database, cf. /D/:

relation scheme	attributes
Suppliers	supplier#, supplier name, supplier location
Parts	part#, part name, color, stock
Shipment	supplier#, part#, quantity

S	S#	SNAME	LOC
	S1	Smith	London
	S2	Jones	Paris
	S3	Blake	Paris

P	P#	PNAME	COLOR	STOCK
	P1	nut	red	London
	P2	bolt	green	Paris
	P3	screw	blue	London

SP	S#	P#	QY
	S1	P1	30
	S2	P1	30
	S2	P2	40
	S3	P4	10



A query "find name of suppliers that currently supply red parts" is expressed in relational algebra as

$$\pi_{\{SNAME\}}(\sigma_{COLOR='red'}(S \bowtie SP \bowtie P)).$$

Here the user relation-wise "navigates" in the database schema by determining the path S, SP, P in the (hypergraph of the) database schema; then the database system automatically generates appropriate operations on tuples in order to calculate the natural join along the path.

The main goal of a universal relation view is to allow queries with no navigation whatever (as far as possible). For this purpose queries are stated only by means of attributes without mentioning the database relations. Instead the user refers to a fictitious universal relation, denoted say by the relation name U, that contains the information of all database relations. For instance, the query above is expressed as

$$\pi_{\{SNAME\}}(\sigma_{COLOR='red'}(U)),$$

or QUEL-like as

retrieve SNAME where COLOR='red'.

The data manipulation language for a universal relation view should be a user-friendly language for querying and updating a database using only the basic concepts of attributes and conditions. It should be as powerful and flexible that it covers the usual applications (in contrast to menue techniques).

In this paper we present our pragmatically oriented approach to a universal relation view. The relationship to other approaches is discussed in Section 8. Our emphasis is on a careful development and justification of our fundamental design decisions. Accordingly we start by stating our general prerequisites (Section 2) and the intended user profile (Section 3). Our definition of the fictitious universal relation is derived both by intuitive considerations (Section 4) and formal investigations (Section 5). Then we treat the problem of possible ambiguities in the interpretation of responses to queries; we introduce a new desirable property of database schemas that we require as additional prerequisite (Section 6). Finally the data manipulation language is outlined (Section 7).

Proofs of theorems are added in an Appendix.

2. General Prerequisites

Our approach is based on a single essential prerequisite: attributes are designed in such a way that they are expressive enough to carry the semantic of the database. This property has been called the universal scheme assumption, see /AP, U2/.

At the moment, no other formal properties (i.e. dependency preservation, lossless join, third normal form, etc.) are necessarily required. However, as it is well-known, such properties are useful in any case in particular for update operations. Furthermore we emphasize that, in order to be pragmatic, we do not require that the database system has facilities for

- nontrivially processing null values,
- enforcing general data dependencies (other than (intrarelational) keys),
- enforcing interrelational constraints (e.g. foreign keys).

For simplicity we shall assume that database relations do not contain any null values.

3. User Profile of the Universal Relation View

Our universal relation view should be used as a view (external schema) on top of a relational database schema (conceptual schema). The intended user profile is given by describing two extreme cases.

The sophisticated user knows the underlying database schema, and he uses the universal relation view for abbreviating queries and updates, thereby avoiding to program join paths by his own.

The casual user knows only (some of) the attributes, more precisely the universal relation scheme, and he expresses his queries essentially by stating the attributes of his interest and appropriate conditions on them (e.g. for denoting selections). He is not aware of the database relations. For him the actions of the database system (semantic of the data manipulation language) are explained in terms of a fictitious universal relation (to be formally defined as the result of a suitable operator applied to the underlying database relations). Often the casual user is not allowed to update the database.

We require that our universal relation view must be consistent with the understanding of <u>all</u> intended users.

In order to support user-friendly man-maschine communication the database system provides users of the universal relation view with reports on actually performed actions, warnings (for instance against possible ambiguities), and error messages.

Some of these communications necessarily rely on the database schema, thus the casual user will not always be able to fully understand them.

We argue, however, that such messages indicate the limit up to which the user concerned should be allowed to employ the view without help of a sophisticated user: this is to ensure

- that responses to queries are informative and not misleading for the user concerned, and
- that updates of database relations are always semantically correct.

4. The Fictitious Universal Relation: Intuitive Approach

Suppose that $D = (R_1,\ldots,R_n)$ is a relational database schema over attribute set U, where $R_i \subset U$ denotes a relation scheme. Let $d = (r_1,\ldots,r_n)$ be an actual database, that is r_i is a relation (finite set of tuples) valid for scheme R_i.

For the purpose of the universal relation view we want to represent the information contained in $(r_1,\ldots,r_n)$ by a single relation u on attribute set U. The most usual way to combine the information contained in several relations is to take the natural join of them.

Looking at the suppliers-parts database we can try to get the combined information by computing the natural join $S \bowtie P \bowtie SP$ yielding

S#	SNAME	LOC	P#	PNAME	COLOR	STOCK	QY
S1	Smith	London	P1	nut	red	London	30
S2	Jones	Paris	P1	nut	red	London	30
S2	Jones	Paris	P2	bolt	green	Paris	40

Unfortunately, the example shows that the join operator is not fully adequate if it happens that there are so-called "dangling" tuples in some database relations or intermediate relations, for instance the tuples

(P3, screw, blue, London) of P,

and

(S3, Blake, Paris, P4, 10) of $S \bowtie SP$.

In order to prevent the occurence of such dangling tuples while performing updates we would need complex facilities to handle indexed null values and interrelational constraints. But, as stated in Section 2, we don´t want to rely on these facilities.

Alternatively we shall define a new operator, the complete join $\overset{n}{\underset{i=1}{\otimes}} r_i$,

according to the following guidelines:

- it also takes care of partial joins $\bowtie_{i \in I} r_i$ for $I \subsetneq \{1,\ldots,n\}$;
- if t is an element of a partial join then, using a single null value $\perp$ as some kind of placeholder, the tuple t_U on U, defined by

  $$t_U(A) := \begin{cases} t(A) & \text{if t is defined for A,} \\ \perp & \text{otherwise,} \end{cases}$$

  is visible in $\overset{n}{\underset{i=1}{\otimes}} r_i$;
- $\overset{n}{\underset{i=1}{\otimes}} r_i$ is free of subsumed tuples;

  that means it does not contain two tuples t_1 and t_2 such that t_1 is identical with t_2 on the nonnull portion of t_1.

In the example the complete join should yield the following relation:

S#	SNAME	LOC	P#	PNAME	COLOR	STOCK	QY
S1	Smith	London	P1	nut	red	London	30
S2	Jones	Paris	P1	nut	red	London	30
S2	Jones	Paris	P2	bolt	green	Paris	40
S3	Blake	Paris	P4	$\perp$	$\perp$	$\perp$	10
$\perp$	$\perp$	$\perp$	P3	screw	blue	London	$\perp$

Notice that we have not taken the partial join that degenerates to a cartesian product. For we want to combine only pieces of information (tuples) which are related to one another in a nontrivial way (having at least one common attribute).

In the discipline of formal methods for database schema design the natural join is also a well-known tool for combining information. In particular, in considering only functional dependencies as semantic constraints, the natural join is proven to be the unique inverse operator of the projection (if the inverse exists at all). Existence of the inverse operator of the projection is usually referred to as lossless join property of the schema.

In computing the complete join we do not exclude lossy (partial) joins in the sense of design theory. For there seems to be no apriori reason to prevent a sophisticated user of the universal relation view from computing a lossy join if the designer of the schema agreed to such a lossy decomposition.

On the other hand, we expect that database schemas without lossy decompositions will be particularly suitable for our universal relation view. For in this case the ratio behind the decomposition (thinking of the database relations r_i as projections of a single relation v: $r_i = \pi_{R_i}(v)$) is directly connected to our concept of defining a universal relation u as the complete join of the database relations,

$u := \bigotimes_{i=1}^{n} r_i$.

Morover, Corollary 3 (in Section 5 below) shows how we could exclude a join from contributing to the fictitious universal relation.

Before now giving the formal definition of the complete join of a database $(r_1,\dots,r_n)$, we emphasize that we must not actually evaluate it for the purpose of our universal relation view. It will be only a theoretical tool for describing the semantic of the data manipulation language.

The following notations are used throughout the paper. Let

$D := (R_1,\dots,R_n)$ a database schema with $n \geq 2$ that is supposed to be connected (considered as hypergraph);

$JP := \{E \mid \emptyset \neq E \subset D$, E connected (as hypergraph)$\}$ the set of join paths in D;

$UE := \{A \mid$ there exists $R_i \in E$ with $A \in R_i\}$ the set of attributes covered by join path E;

$\bowtie_E d := \bowtie_{R_i \in E} r_i$ for join path E and database $d=(r_1,\dots,r_n)$ the partial join of d with respect to E;

$\varkappa(r) := \{t_U \mid t \in r\}$ the U-expansion of relation r;

$t_1 \leq t_2$: iff t_1, t_2 are tuples on U such that for all $A \in U$
$t_1(A) \neq \perp \Rightarrow t_1(A) = t_2(A)$;

maxtuple $(v) := \{t \mid t \in v$, and for all $t' \in v : t \leq t' \Rightarrow t = t'\}$.

Definition:

The mapping
$\otimes : \{d \mid d$ database for schema $D\} \rightarrow \{u \mid u$ relation on $U\}$,

$\otimes\, d :=$ maxtuple $\left(\bigcup_{E \in JP} \varkappa(\bowtie_E d) \right)$

is called complete join (for D).

The complete join evaluates all noncartesian partial joins, interprets the resulting tuples as tuples on U by formally introducing null values, and finally removes redundancy.

5. The Fictitious Universal Relation: Formal Approach

The basic query of our data manipulation language is

retrieve X,

where X is a set of attributes.

For the casual user, obviously the effect of this query should be

$\pi_X(urv(d))$,

where urv(d), universal relation view of d, is the fictitious universal relation that is computed from the actual database $d=(r_1,\dots,r_n)$.

More formally, urv is a mapping

urv: $\{d \mid d$ database for schema $D\} \rightarrow \{u \mid u$ relation on $U\}$,

such that urv(d) contains only values occuring in d and, if necessary, the null value $\perp$.

(For the moment, urv(d) is not necessarily the complete join, but see Theorem 4 below.)

Since the null value $\perp$, as introduced in Section 4, does not carry any real information, we decided to define

π_X as the total projection on X yielding only tuples without nulls.

For the sophisticated user, the query is intended to be an abbreviation of a request to evaluate one or several join paths. Thus the database system must relate the attribute set X with a set of join paths covering X.

More formally the database system needs a specification of a join path function

$$jp: \{X \mid X \subset U\} \rightarrow \{Q \mid Q \subset JP\} \quad \text{such that} \quad E \in jp(X) \Rightarrow X \subset \cup E;$$

based on jp the query has the effect of computing the relation

$$\bigcup_{E \in jp(X)} \pi_X(\bowtie_E d) .$$

Of course, we can always optimize jp by actually taking only the minimal (with respect to $\subset$) elements of jp(X), i.e. for all X we have

$$\bigcup_{E \in jp(X)} \pi_X(\bowtie_E d) = \bigcup_{\substack{E \in jp(X) \\ E \text{ minimal in } jp(X)}} \pi_X(\bowtie_E d) .$$

Since both understandings of the query should be consistent, we have to require the following condition on urv and jp:

property 1 [consistent understanding]

for all $d = (r_1,\dots,r_n)$, for all $X \subset U$:

$$\pi_X(urv(d)) = \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d) .$$

Moreover, for a given join path function jp the fictitious universal relation urv(d) should not contain any spurious information:

property 2 [minimality]

for all $d = (r_1,\dots,r_n)$: urv(d) is the minimum relation satisfying the property of consistent understanding, that means urv(d) is the minimum element of

$$\{u \mid \text{for all } X \subset U: \pi_X(u) = \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d)\},$$

where $u_1 \leq u_2$: iff $u_1 \subset u_2$ and for all $t_2 \in u_2$ there exists $t_1 \in u_1$ such that $t_2 \leq t_1$.

Given a join path function jp for the sophisticated user (that is a detailed specification of abbreviations provided by the universal relation view) we are asking whether property 1 and property 2 are satisfiable by some mapping urv. We present the following fundamental theorems:

Theorem 1

Let jp be a join path function and d a database. Then the following assertions are equivalent:

a) There exists u such that for all $X \subset U$:

$$\pi_X(u) = \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d) .$$

b) For all X, Y with $X \subset Y \subset U$:

$$\pi_X\Big[\bigcup_{E \in jp(Y)} \pi_Y(\bowtie_E d)\Big] \subset \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d).$$

[Note: For the total projection we have $\pi_X(\pi_Y(r)) \subset \pi_X(r)$ if $X \subset Y$, but in general not equality!]

Theorem 2

Under the suppositions of Theorem 1, if the assertions of Theorem 1 hold then there exists a minimum relation u satisfying assertion a); moreover

$$u = \text{maxtuple}\Big[\bigcup_{X \subset U} \bigcup_{E \in jp(X)} \varkappa(\pi_X(\bowtie_E d))\Big].$$

In order to guarantee property 1, consistent understanding, for an arbitrary join path function jp we would need complex facilities for maintaining the interrelational constraints given in assertion b) of Theorem 1.

However, there is a natural condition on jp under which assertion b) always holds for any database d:

Corollary 3

Let jp be a join path function such that for all $X,Y \subset U$, for all $E \in JP$:
$X \subset Y$ and $E \in jp(Y) \Rightarrow$
there exists $F \subset E$ with $F \in jp(X)$.

Then for all $d = (r_1,\dots,r_n)$

$$\text{maxtuple}\Big[\bigcup_{X \subset U} \bigcup_{E \in jp(X)} \varkappa\,(\pi_X(\bowtie_E d))\Big]$$

is the minimum relation u such that for all $X \subset U$:

$$\pi_X(u) = \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d).$$

As discussed in Section 4, the fictitious universal relation urv(d) should make visible all partial joins $\bowtie_E d$; thus we additionally require for urv property 3 [visibility of partial joins]

for all $d=(r_1,\ldots,r_n)$, for all $E \in JP$:

$$\bowtie_E d \subset \pi_{UE}(urv(d)).$$

The next theorem states that properties 1, 2, and 3 already completely determine jp and urv.

Theorem 4

The following assertions are equivalent:
a) jp and urv satisfy properties 1, 2, and 3.
b) For all $d=(r_1,\ldots,r_n)$, for all $X \subset U$:
  i) $jp(X) = \{E \mid E \in JP, \text{ and } X \subset UE\}$ up to optimization, and
  ii) $urv(d) = \otimes d$.

6. Unambiguous partial joins and characteristic attributes

Both the intuitive approach (Section 4) and the formal approach (Section 5) support our fundamental decision to define the fictitious universal relation by

$$urv(d) := \otimes d .$$

This implies that the semantic, denoted by the evaluation function eval, of the basic query is given by

$$eval(d;\ \underline{retrieve}\ X) := \pi_X(\otimes d) = \bigcup_{\substack{E \in JP \\ X \subset UE}} \pi_X(\bowtie_E d).$$

In general there will be more than just one minimal join path E with $X \subset UE$, say $E_1,\ldots,E_k$. Then the interpretation of a tuple $t \in eval(d;\ \underline{retrieve}\ X)$, as seen by the user of the universal relation view, will be ambiguous: it means

$$t \in \pi_X(\bowtie_{E_1} d) \quad \text{or} \quad \ldots \quad \text{or} \quad t \in \pi_X(\bowtie_{E_k} d).$$

This ambiguity is caused by two reasons.

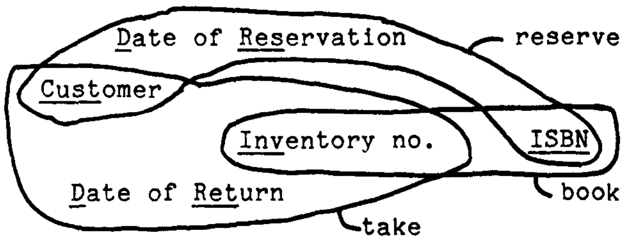
On the one hand, X may be a subset of several base relations, for instance in our suppliers-parts database we have

$$eval(d, \underline{retrieve}\ P\#) = \pi_{\{P\#\}}(P) \cup \pi_{\{P\#\}}(SP) = \begin{array}{c} P\# \\ \hline P1 \\ P2 \\ P3 \end{array} \cup \begin{array}{c} P\# \\ \hline P1 \\ P2 \\ P4 \end{array} .$$

In this case our semantic provides a good answer, namely all part numbers the system knows about.

On the other hand, the ambiguity may stem from the existence of some kind of cycle in the database schema (considered as hypergraph), see, for instance, /AP, F/. In general, this situation seems unavoidable since nontrivial applications usually have to model inherently cyclic features of the real world.

For instance, in a library
- a copy of a book is identified by an inventory number, and its content is described by an ISBN entry;
- customers can take out a copy or, if all copies of the book are currently taken out, make a reservation.



Obviously there are two relationships between customers and copies; accordingly we have

$$eval(d;\ \underline{retrieve}\ cust,inv) = \pi_{\{cust,inv\}}(take) \cup \pi_{\{cust,inv\}}(reserve \bowtie book).$$

In dealing with ambiguity, we argue for the following two requirements:
- In the special case that X is equal to the set of attributes covered by some join path E, i.e. X=UE, there should be no essential ambiguity. For the sophisticated user should be able to get the exact value of $\bowtie_E d$ by means of the query $\underline{retrieve}$ UE.
- In the other cases it should always be possible to remove the ambiguity by specifying additional attributes. In the library example we can take

  $\underline{retrieve}$ Cust,Inv $\underline{where}$ DRet=DRet,

  respectively

  $\underline{retrieve}$ Cust,Inv $\underline{where}$ DRes=DRes.

  (See Section 7 below for an explanation of nonbasic queries.)

Thus in any case the (more sophisticated) user should be able to get a precise unambiguous result.

We shall formalize the first requirement by property 4 below and subsequently show how this property can be efficiently guaranteed by property 5 in such a way that the second requirement is also satisfied:

property 4 [unambiguous visibility of partial joins]

for all $d=(r_1,\dots,r_n)$, for all $E \in JP$:

$$\bowtie_E d = \pi_{UE}(\otimes d).$$

The condition of property 4 can be equivalently stated as follows:

Theorem 5

Let $d=(r_1,\dots,r_n)$. Then the following assertions are equivalent:

a) For all $E \in JP$: $\bowtie_E d = \pi_{UE}(\otimes d)$.

b) For all $E \in JP$, for all $R_i \in D$:
$R_i \subset UE \Rightarrow \pi_{R_i}(\bowtie_E d) \subset r_i$.

As assertion b) of Theorem 1, in general the assertion b) above requires to maintain a complex interrelational constraint.

However, we found a natural condition on the database schema D under which assertion b) always holds for any database; furthermore essentially there does not exist any other possibility to guarantee property 4 on the conceptual level:

property 5 [unambiguous database schema]

for all $E \in JP$, for all $R_i \in D$:
$R_i \subset UE \Rightarrow R_i \in E$.

Theorem 6

1. Property 5 implies property 4.

2. Suppose that there are no interrelational constraints for database schema D; furthermore suppose that the intrarelational constraints admit some natural simple databases (see the proof). Then property 4 implies property 5.

Property 5 can be easily tested:

Theorem 7

The following assertions are equivalent:

a) [unambiguous database schema]
For all $E \in JP$, for all $R_i \in D$:
$R_i \subset UE \Rightarrow R_i \in E$

b) [existence of characteristic attributes]
For all $R_i \in D$:
$D \setminus \{R_i\}$ connected (considered as hypergraph)
$\Rightarrow$ there exists $A \in R_i \setminus U(D \setminus \{R_i\})$.

An attribute $A \in R_i \setminus U(D \setminus \{R_i\})$ is called a characteristic attribute of relation scheme R_i. In expressing queries it can serve as a substitute for the name of the relation scheme. We believe that the concept of characteristic attributes is fundamental for the design of database schemas that satisfy the universal scheme assumption. According to our experience, good designs nearly always possess characteristic attributes. On the other hand, if a tentative design does not, the possible sources of ambiguity can be detected using assertion b) above and subsequently eliminated by introducing a suitable new attribute for the relation concerned.

As announced at the beginning of this section, a user of the universal relation view can employ property 5 to remove ambiguities in the interpretation of a query, say retrieve X. As an example, assume that there are two minimal join paths $E_1 \neq E_2$ covering X. Then, for an unambiguous database schema there must exist attributes A_1, respectively A_2, with $A_1 \in UE_1 \setminus UE_2$, respectively $A_2 \in UE_2 \setminus UE_1$. (Otherwise, supposing indirectly $UE_1 \subset UE_2$ we could derive $E_1 \subset E_2$ using property 5, a contradiction.)

Thus the user can "navigate" through the schema, i.e. determine exactly that join path that he wishes to be evaluated, by restating his query as retrieve X_1, respectively retrieve X_2, where X_i is an appropriate set with $X \cup \{A_i\} \subset X_i \subset UE_i$.

Referring to our motivation for a universal relation view, namely querying "without navigation", we can now clarify this point:

in general the user need not to navigate explicitly; however in those cases for that some kind of navigation is inherently necessary, due to the cyclic nature of an application, he still can "navigate" using the expressive power of characteristic attributes.

7. The Data Manipulation Language

The semantic of our data manipulation language is based on the fundamental decisions discussed in the previous sections:
- to consistently serve users ranging from being casual up to sophisticated (Section 3),
- to define the fictitious universal relation as the complete join of the database (Section 4 and 5), and
- to suppose an underlying unambiguous database schema (Section 6).

For the sake of brevity, in the present report we only outline the basic constructs of our language. A detailed exposition will be given separately /BB/.

A query has the general form

$q \equiv$ retrieve attribute list
where condition.

Here the attribute list is constructed from qualified attributes (of the form: tuple_variable.attribute), and the condition is a boolean combination (for the time of writing: without negation) of elementary selection conditions (es).

The semantic is defined as follows:

i) If q is a basic conjunctive query, i.e. it contains only one tuple variable t and its condition is a conjunction of elementary selection conditions,

$q \equiv$ retrieve $t.A_1,\dots,t.A_k$
where $es_1 \wedge \dots \wedge es_l$,

then

$$eval(d;q) := \pi_{\{t.A_1,\dots,t.A_k\}}(\sigma_{es_1\wedge\dots\wedge es_l}(rename(t, \otimes d)))$$

where rename(t,u) is a copy of relation u, where each attribute is qualified by the tuple variable t, i.e. instead of $A \in U$ we take t.A.

Example:

"Find name of suppliers that currently supply part P1":

$q_1 :\equiv$ retrieve t.SNAME
where t.P# = 'P1'.

In this case, since there is only one tuple variable, the tuple variable can be omitted at all and thus we don't need the renaming function:

$$eval(d;q_1) = \pi_{\{SNAME\}}(\sigma_{P\#='P1'}(\otimes d))$$

yielding

SNAME
Smith
Jones.

ii) If q is a join conjunctive query, i.e. it contains tuple variables $t_1,\dots,t_a$ with $a \geq 2$ and its condition is a conjunction of elementary selection conditions,

$q \equiv$ retrieve $t_{i_1}.A_1,\dots,t_{i_k}.A_k$
where $es_1 \wedge \dots \wedge es_l$,

then

$$eval(d;q) := \pi_{\{t_{i_1}.A_1,\dots,t_{i_k}.A_k\}}(\sigma_{es_1\wedge\dots\wedge es_l}(\overset{a}{\underset{j=1}{\times}} rename(t_j, \otimes d))).$$

Example:

"Find name of suppliers and name of parts such that location of supplier is equal to stock of part"
(Note: the suppliers need not to supply this part currently):

$q_2 :\equiv$ retrieve t_1.SNAME, t_2.PNAME
where $t_1.LOC = t_2.STOCK$;

$$eval(d;q_2) = \pi_{\{t_1.SNAME,t_2.PNAME\}}(\sigma_{t_1.LOC=t_2.STOCK}(rename(t_1, \otimes d) \times rename(t_2, \otimes d)))$$

yielding

t_1.SNAME	t_2.PNAME
Smith	nut
Smith	screw
Jones	bolt
Blake	bolt.

iii) If q is not conjunctive, then its condition c is equivalent to a (unique) condition in disjunctive normal form $c_1 \vee \dots \vee c_b$ where each c_i is a conjunction of elementary selection conditions,

$q \equiv$ retrieve attribute list
where $c_1 \vee \dots \vee c_b$;

then

$$eval(d;q) := \bigcup_{i=1}^{b} eval(d; \text{retrieve attribute list where } c_i).$$

Example:

"Find name of suppliers that currently supply part P4 or nuts":

q_3:≡ retrieve t.SNAME
    where t.P#='P4'
        or t.PNAME='nut',

eval(d;q_3) =
  eval(d; retrieve t.SNAME
        where t.P# ='P4')
$\cup$ eval(d; retrieve t.SNAME
        where t.PNAME='nut')

$$= \pi_{\{SNAME\}}(\sigma_{P\#='P4'}(\otimes d)) \cup \pi_{\{SNAME\}}(\sigma_{PNAME='nut'}(\otimes d)),$$

yieding	SNAME
	Blake
	Smith
	Jones.

Expansion in disjunctive normal form should indicate how the selection operator σ works on tuples with null values, see also the following note on optimizations.

Of course, in general we don't have to compute the whole universal relation $\otimes$ d. As an optimization, it suffices to proceed as follows instead:

for each tuple variable t of conjunctive query q we determine the attribute set $W_t := \{A \mid t.A \text{ occurs in } q\}$; then we can replace $\otimes$ d by

$$\pi_{W_t}(\otimes d) = \bigcup_{\substack{E \in JP \\ W_t \subset \cup E}} \pi_{W_t}(\bowtie_E d),$$

where we actually need to consider only the minimal such E's.

As with other views, updating of the universal relation view is possible only in rather restricted form (cf. /DB, Si/). The most basic update statements are

insert t , respectively delete t

where t is a tuple defined for some attribute set $W \subset U$.

Our semantic of update statements is guided by the following principles:
a) The user should be able to reconstruct the original database after an erroneous update.
b) The effect of an update should always be visible by a subsequent query of the basic form retrieve W.
c) If an update was performable only on the basis of some arbitrary decision, probably obscure for a user, then it should be rejected at all.

The effect of insert t is roughly described as follows:

```
begin transaction
   failure:= false;
   E:= {$R_i \mid R_i \in D, R_i \subset W$};
   if $\cup E \subsetneq W$ or E not connected
   then (error message ; exit);
   for $R_i \in E$ do
      (try to insert subtuple t[$R_i$] into
       $r_i$ ;
       if insertion rejected [because of
          violation of constraints]
       then (prepare error message;
             failure:= true)
       else
          if t[$R_i$] was newly inserted
             in $r_i$
          then prepare success message
      );
   if failure then undo else commit
end transaction.
```

The effect of delete t is decribed by

```
   if W = $R_i$ for some $R_i \in D$
   then try to delete t from $r_i$
   else error message.
```

8. Relationship to other Approaches

Recently many different approaches to a universal relation view have been presented /O; KU, K, MU, FMU, U2; MW, MRSSW; MS, Sa; L/.

By assessing their advantages and disadvantages we adopted many useful ideas from previous approaches. However, our proposal as a whole is essentially different from all of them.

The most advanced projects are due to Korth/Ullman et al. /KU, K, MU, FMU, U2/, Maier/Warren et al. /MW, MRSSW/ and Sagiv et al. /MS, Sa/.

The query interpretation algorithm of the System/U by Korth/Ullman uses an optimization technique which assumes a globally consistent database (i.e. without dangling tuples) satisfying global semantic constraints. The proposed update procedures to maintain such a database are based on extensive usage of indexed null values and enforcement of complex semantic constraints by applying tableau chase techniques on actual relations (instead of schemes). As far as we see, these features seem to be not efficiently implementable. However Korth/Ullman do not insist that the database is indeed free of dangling tuples, rather they argue in favor of their approach that the user will expect to get some kind of "simplest" response.

Maier/Warren, as already anticipated in System/U, employ the notion of "objects" as basic construct for navigation. They suggest to actually define the database schema in terms of both relation schemes (there called "associations") and objects. In applications objects are intended to model relevant real world concepts as they are referred to in natural languages, for instance. In contrast, our approach relies only on a conventional relational database schema. On the other hand, our notion of a join path function is related to the concept of objects, as our conditions for consistent understanding, respectively unambiguity (assertion b) of Theorem 1, respectively assertion b) of Theorem 5) are related to the Maier/Warren "containment condition" for objects.

The spirit of Sagiv´s approach seems to be closest to our´s: he tries to avoid null values and is looking for conditions that allow to efficiently compute projections of the fictitious universal relation without knowing it explictly. However, his fictitious universal relation, the so-called "representative instance", is different from our´s.

Finally we note that our complete join is in the spirit of the operation "maxtrav" (maximal traversal) of Lien /L/.

After preparing the first version of this paper we learnt from further work on universal relation views /CK, B, Second edition of U1, MUV, DMS/.

Carlson and Kaplan /CK/ present an early study on query languages based on attributes. They propose a method to select an appropriate "access path" on the basis of the database system´s information about functional dependencies and interrelational constraints.

Babb /B/ describes how to construct a single "joined normal form" relation that is intended to model a database application and that can be stored in one Content Addressable File Store. Furthermore a suitable data manipulation language has been developed. Although Babb´s work is directed to employ the power of the Content Addressable File Store whereas we intend to use a conventional relational database system, it turns out there are some similarities between these approaches.

In the second edition of his book /U1/ Ullman presents an actual and comprehensive treatment of the System/U.

Maier, Ullman, and Vardi /MUV/ study the relationship between the approach taken for the System/U and by Maier/Warren and the "representative instance" approach of Sagiv.

Finally D´Atri, Moscarini, and Spyratos /DMS/ define a notion of unambiguous context -a context corresponds to a join path- and relate this notion to the relationship uniqueness assumption and to γ-acyclicity.

9. Conclusion

On the basis of realistic prerequisites and a determination of te intended user profile we carefully developped our fundamental decisions for a universal relation view: consistent understanding for all users; complete join as defining operator for the fictitious universal relation; unambiguous database schemas. The data manipulation language is outlined.

We are currently involved in implementing a prototype version of our approach.

Acknowledgments:

We thank the members of the Projektgruppe DURST (twelve students of computer science at University of Dortmund joining the project entitled DURST, "Datenbank mit Universalrelation-Schnittstelle") for many discussions and contributions that helped to clarify the concepts presented in this paper:
M. Brocke, R. Burhop, B. Convent, D. Ellinghaus, J. Kalinski, M. Kramer, R. Liebold, H. Michaelis, U. Räsch, S. Richter, L. Schnetgöke, H. Stiefeling.

Furthermore we are grateful to the referees for constructive comments on the first version of this paper and to colleagues in Rome, Pisa, and Paderborn for helpful discussions.

References:

/AP/ Atzeni, P., Parker, D.S., Assumptions in relational database theory, Proc. 1st ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1982, pp. 1-9

/B/ Babb, E., Joined Normal Form: A Storage Encoding for Relational Databases, ACM Transactions on Database Systems, Vol.7, No.4, Dec. 1982, pp. 588-614

/BB/ Biskup, J., Brüggemann, H.H., Eine Datenbanksprache für eine Universalrelation-Sicht, unpublished manuscript, 1983

/CK/ Carlson, C.R., Kaplan, R.S., A Generalized Access Path Model and its Application to a Relational Data Base System, Proc. ACM SIGMOD Int. Conference on Management of Data, 1976, pp. 143-154

/D/ Date, C.J., An introduction to Database Systems, Addison-Wesley, Reading, 3rd edition, 1981

/DMS/ D'Atri, A., Moscarini, M., Spyratos, N., Answering Queries in Relational Databases, Proc. ACM SIGMOD Database Week, May 1983

/DB/ Dayal, U., Bernstein, P.A., On the updatability of relational views, Proc. 4th Intern. Conf. on Very Large Data Bases, 1978, pp. 368-377

/F/ Fagin, R., Degrees of acyclicity for hypergraphs and relational database schemes, to appear J. ACM

/FMU/ Fagin, R., Mendelzon, A.O., Ullman, J.D., A simplified universal relation assumption and its properties, ACM Trans. on Database Systems, Vol.7, No.3, Sept. 1982, pp. 343-360

/K/ Korth, H.F., System/U: a progress report XP2 Workshop on Relational Database Theory, State College, 1981

/KU/ Korth, H.F., Ullman, J.D., System/U: a database system based on the universal relation assumption, XP1 Workshop on Relational Database Theory, Stony Brook, 1980

/L/ Lien, Y.E., On the equivalence of database models, Journal ACM, Vol.29, No.2, 1982, pp.333-362

/MRSSW/ Maier, D., Rozenshtein, D., Salveter, S., Stein, J., Warren, D.S., Towards logical data indepen dence: a relational query language without relations, Proc. ACM SIGMOD Int. Conference on Management of Data, 1982, pp. 51-60

/MU/ Maier, D., Ullman, J.D., Maximal objects and the semantics of universal relation databases, ACM Trans. on Database Systems, Vol.8, No.1, March 1983, pp. 1-14

/MUV/ Maier, D., Ullman, J.D., Vardi, M.Y., The Equivalence of Universal Relation Definitions, STAN-CS-82-940, Dept. of Computer Science, Stanford, 1982

/MW/ Maier, D., Warren, D.S., Specifing connections for a universal relation scheme database, Proc. ACM SIGMOD Int. Conference on Management of Data, 1982, pp. 1-7

/MS/ McCure Kuck, S., Sagiv, Y., A universal relation database system implemented via the network model, Proc. 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, 1982, pp.147-157

/O/ Osborn, S.L., Towards a universal relation interface, Proc. 5th Intern. Conference on Very Large Data Bases, 1979, pp. 52-60

/Sa/ Sagiv, Y., Can we use the universal instance assumption without using nulls?, Proc. ACM SIGMOD Int. Conference on Management of Data, 1981, pp. 108-120

/Si/ Siklossy, L., Updating views: a constructive approach, Preprints Workshop Logical Bases for Data Bases, Toulouse, 1982

/U1/ Ullman, J.D., Principles of Database Systems, Computer Science Press, Potomac, 1980; 2nd revised edition 1983

/U2/ Ullman, J.D.,
The U.R. strikes back,
Proc. 1st ACM SIGACT-SIGMOD
Symp. on Principles of Database
Systems, 1982, pp. 10-22

Appendix

Proof of Theorem 1:

(a) $\Rightarrow$ (b) :

$X \subset Y \subset U$

$\Rightarrow \pi_X(\pi_Y(u)) \subset \pi_X(u)$ (by definition of total projection)

$\Rightarrow \pi_X[\bigcup_{E \in jp(Y)} \pi_Y(\bowtie_E d)] \subset \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d)$

(by a)

(b) $\Rightarrow$ (a) :

Define $u := \bigcup_{X \subset U} \bigcup_{E \in jp(X)} \varkappa[\pi_X(\bowtie_E d)]$

(1) $\bigcup_{E \in jp(X)} \pi_X(\bowtie_E d) \subset \pi_X(u)$ for all $X \subset U$ (by definition of u)

(2) $\pi_X(u) = \pi_X[\bigcup_{Y \subset U} \bigcup_{E \in jp(Y)} \varkappa[\pi_Y(\bowtie_E d)]]$

(by definition of u)

$= \pi_X[\bigcup_{X \subset Y \subset U} \bigcup_{E \in jp(Y)} \varkappa[\pi_Y(\bowtie_E d)]]$

(for π_X is the total projection)

$= \bigcup_{X \subset Y \subset U} \pi_X[\bigcup_{E \in jp(Y)} \pi_Y(\bowtie_E d)]$

($\varkappa$ superfluous)

$\subset \bigcup_{X \subset Y \subset U} \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d)$

(by b)

$= \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d)$

Proof of Theorem 2:

Let URV(d) :=

$\{u' \mid \forall X \subset U : \pi_X(u') = \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d)\}$

We prove that $u \leq u'$ for all $u' \in URV(d)$, i.e.

$u' \in URV(d) \Rightarrow$ (1) $\forall t_2 \in u' \exists t_1 \in u : t_2 \leq t_1$, and
(2) $u \subset u'$.

Since $u, u' \in URV(d)$, we have for all $X \subset U$:

$$\pi_X(u) = \pi_X(u') \qquad (*)$$

(1) Let $t_2 \in u'$, $X := \{A \mid t_2[A] \neq \bot\}$.

$\Rightarrow t_2[X] \in \pi_X(u')$
(by definition of total projection)

$\Rightarrow \exists t_1 \in u : t_1[X] = t_2[X]$
(by (*))

$\Rightarrow \exists t_1 \in u : t_2 \leq t_1$
(by definition of X)

(2) Assume $\exists t_1 \in u \setminus u'$
Let $X := \{A \mid t_1[A] \neq \bot\}$.

$\Rightarrow \forall t_2 \in u' : t_1[X] \neq t_2[X]$

$\Rightarrow \pi_X(u) \neq \pi_X(u')$

This is a contradiction to (*).

Proof of Corollary 3:

Let $X \subset Y$.

$\pi_X[\bigcup_{E \in jp(Y)} \pi_Y(\bowtie_E d)]$

$= \bigcup_{E \in jp(Y)} \pi_X(\pi_Y(\bowtie_E d))$

$= \bigcup_{E \in jp(Y)} \pi_X(\bowtie_E d)$

$\subset \bigcup_{F \in jp(X)} \pi_X(\bowtie_F d)$ (by supposition and $F \subset E \Rightarrow \pi_X(\bowtie_E d) \subset \pi_X(\bowtie_F d)$)

Then apply Theorem 1 and 2.

Proof of Theorem 4:

(a) $\Rightarrow$ (b) :

(i) "$\subset$":

$\forall E \in JP$: $\bowtie_E d \subset \pi_{UE}(urv(d))$ (by property 3)

$\Rightarrow \forall E \in JP$:

$[X \subset UE \Rightarrow \pi_X(\bowtie_E d) \subset \pi_X(\pi_{UE}(urv(d)))]$

$\subset \pi_X(urv(d))$

$\Rightarrow \bigcup_{\substack{E \in JP \\ X \subset UE}} \pi_X(\bowtie_E d) \subset \pi_X(urv(d))$

"$\supset$":

$\pi_X(urv(d)) = \bigcup_{E \in jp(X)} \pi_X(\bowtie_E d)$ (by property 1)

$\subset \bigcup_{\substack{E \in JP \\ X \subset UE}} \pi_X(\bowtie_E d)$ (by definition of jp)

Hence we have

$\pi_X(urv(d)) = \bigcup_{\substack{E \in JP \\ X \subset UE}} \pi_X(\bowtie_E d)$

and so (i) holds.

(ii) urv(d) =

maxtuple $[\bigcup_{X \subset U} \bigcup_{\substack{E \in JP \\ X \subset UE}} \varkappa(\pi_X(\bowtie_E d))]$ =

(by Thm. 2 and (i) above)

maxtuple $[\bigcup_{E \in JP} \varkappa(\bowtie_E d)]$ =

(by the definition of the complete join)

$\otimes$ d

(b) $\Rightarrow$ (a) :

property 1:

$\pi_X(\otimes d)$ =

$\pi_X(\text{maxtuple}[\bigcup_{E \in JP} \varkappa(\bowtie_E d)])$ =

(by definition of $\otimes$)

$\pi_X(\bigcup_{E \in JP} \varkappa(\bowtie_E d))$ =

$\bigcup_{\substack{E \in JP \\ X \subset UE}}$ (for π_X is the total projection)

$\bigcup_{\substack{E \in JP \\ X \subset UE}} \pi_X(\bowtie_E d)$ =

$\bigcup_{E \in jp(X)} \pi_X(\bowtie_E d)$ (by (b)(i))

property 2:

$\otimes$ d =

maxtuple$(\bigcup_{E \in JP} \varkappa(\bowtie_E d))$ =

maxtuple$(\bigcup_{X \subset U} \bigcup_{\substack{E \in JP \\ X \subset UE}} \varkappa(\pi_X(\bowtie_E d)))$ =

maxtuple$(\bigcup_{X \subset U} \bigcup_{E \in jp(X)} \varkappa(\pi_X(\bowtie_E d)))$

(by (b)(i))

Then property 2 follows from Corollary 3

property 3:

$\pi_{UE}(\otimes d) \supset \bowtie_E d$ (by definition of $\otimes$)

Proof of Theorem 5:

(a) $\Rightarrow$ (b):

$\pi_{R_i}(\bowtie_E d) \subset \pi_{R_i}(\otimes d)$

$= \bowtie_{\{R_i\}} d$ (by (a) with $E = \{R_i\} \in JP$)

$= r_i$

(b) $\Rightarrow$ (a):

"$\subset$": by property 3

"$\supset$": Let $E \in JP$.

$$\pi_{UE}(\otimes d) = \bigcup_{\substack{F \in JP \\ UE \subset UF}} \pi_{UE}(\bowtie_F d) \quad \text{(by Theorem 4)}$$

Let $t \in \pi_{UE}(\otimes d)$

$\Rightarrow \exists F \in JP : t \in \pi_{UE}(\bowtie_F d)$ and $UE \subset UF$.

Then

$R_i \in E \Rightarrow R_i \subset UE \subset UF$

$\Rightarrow \pi_{R_i}(\bowtie_F d) \subset r_i$ (by (b))

$\Rightarrow \{t[R_i]\} = \pi_{R_i}(\{t\})$

$\subset \pi_{R_i}(\bowtie_F d) \subset r_i$

hence $t = t[UE] \in \bowtie_E d$

<u>Proof of Theorem 6</u>:

(1) Let $E \in JP$.

$R_i \subset UE \Rightarrow R_i \in E$ (by property 5)

$\Rightarrow \pi_{R_i}(\bowtie_E d) \subset r_i$

(by definition of join)

$\Rightarrow \bowtie_E d = \pi_{UE}(\otimes d)$ (by Theorem 5)

(2) We shall prove the contraposition:
Let w.l.o.g. $E \in JP$, $R_1 \in D$ with
$R_1 \subset UE$ and $R_1 \notin E$.
Let $r_1 := \emptyset$ and $r_i := \{t[R_i]\}$ for
$i=2,\ldots,n$ for some tuple t on U.
Let $d := (r_1,\ldots,r_n)$.
We require that the intrarelational constraints admit that the r_i are valid relations for R_i. Then d is a valid database for schema D.
But we have

$\bowtie_E d = \{t[UE]\}$ (by construction of d and $R_1 \notin E$),

$\pi_{R_1}(\bowtie_E d) = \{t[R_1]\}$ (by t is nullfree and $R_1 \subset UE$),

and $\bowtie_{\{R_1\}} d = r_1 = \emptyset$ (by construction of d)

So property 4 does not hold for d.

<u>Proof of Theorem 7</u>:

(a) $\Rightarrow$ (b): Let $D \setminus \{R_i\}$ be connected.
Assume indirectly $R_i \subset U(D \setminus \{R_i\})$

$\Rightarrow R_i \in D \setminus \{R_i\}$ (by (a) with $E = D \setminus \{R_i\} \in JP$)

This is a contradiction.

(b) $\Rightarrow$ (a): Let $R_i \subset UE$.

case 1: $D \setminus \{R_i\}$ is connected:

$\Rightarrow \exists A \in R_i \setminus U(D \setminus \{R_i\})$ (by (b))

$\Rightarrow R_i \in E$ (else $A \notin UE$ and $R_i \not\subset UE$)

case 2: $D \setminus \{R_i\}$ is not connected:

Since D is connected, R_i connects all the maximal components C_j of $D \setminus \{R_i\}$. (*)

Assume indirectly $R_i \notin E$.

$\Rightarrow E \subset D \setminus \{R_i\}$ (since $R_i \notin E$)

$\Rightarrow \exists C_j : E \subset C_j$ (since E connected and C_j max. components)

$\Rightarrow \exists k \neq j : \exists A : A \in R_i \cap UC_k$ (by (*))

$\Rightarrow \exists A \in R_i : A \notin UC_j \supset UE$

(since C_j, C_k not connected, $E \subset C_j$)

$\Rightarrow R_i \not\subset UE$.

This is a contradiction.