International Journal of Computer and Information Sciences, Vol. 9, No. 4, 1980

WCRL: A Data Model Independent Language for Database Systems

Sudhir K. Arora[1] and K. C. Smith[2]

Received July 1979; revised April 1980

The need for data model independent languages for database systems has become apparent in recent years. They can be used for the conceptual level of a database system, for communication in a distributed database system, for data restructuring, and so on. This paper proposes a language, WCRL, to fill this need and compares it with the very few other languages which have been developed almost concurrently to fill the same need.

KEY WORDS: Data retrieval; database language; data model independent language.

1. INTRODUCTION

Several database languages have appeared in the literature—QUEL,[13] SEQUEL,[5] QUERY BY EXAMPLE,[16] SYNGLISH,[9] etc. All these languages deal with a particular data model—Relational, Network, Hierarchical, etc. However, there is a need for a database language which can apply equally well to at least the three major data models. Such a language may be useful in the following situations.

1. The ANSI/X3/SPARC report[1] requires a database management system to have three levels—the external, the conceptual, and the internal. At the external level, several users of the database should be able to use different data models. This is called logical data independence. Logical data

[1] Department of Electrical and Computer Engineering, McMaster University, Hamilton L8S 4L7, Ontario, Canada.
[2] Department of Electrical Engineering, University of Toronto, Toronto, Ontario, Canada.

0091-7036/80/0800-0287$03.00/0

independence makes it necessary to have a common language at the condeptual level for different data models at the external level.

2. In a distributed computer network, we may have several databases, each with its own model of data. A common data model independent language could serve as a means of communication between the databases.

3. A data model independent language could be useful for transforming data from one model to another.

Tsichritzis presented a data model independent language, LSL, in 1976.[14] Since then there have been only three other languages for tish purpose—FQL in 1979,[4] QUEST in 1979,[8] and WCRL in 1979, which is presented in this paper.

Further, in the literature set processing capabilities have appeared in several languages. QUEL has SET and AGGREGATE clauses; SEQUEL has a GROUP-BY clause; Query by Example incorporates subsetting operations; SYNGLISH provides special key words such as SOME, ALL, or ONLY to obtain subsetting; Algebra of Quotient Relations[7] has set comparison operators. These trends are there because set processing offers efficient query processing in a database. Also, Pecherer[10] and Smith and Chang[12] suggest the use of conditional expressions involving more than one comparison in a relational language.

This paper introduces a language based on a data structure, called well connected relation (WCR), which has been studied by us elsewhere.[3] The language applies equally well to the Network, the Relational, and the Hierarchical data models. The language is algebraic in nature but allows navigation through a Network database. It combines efficient set processing with complex conditional expressions in the queries. The language is complete in the relational sense of Codd[6] and also allows the least fixed point operations of Aho and Ullman.[2] In this sense it is more powerful thna relational algebra.

Section 2 of this paper introduces WCR's. Section 3 presents the WCR language which has a high level and a lpw level part. Section 4 presents a generalized Network model and examples of queries in WCRL on both the Network and the Relational models. The Hierarchical model is a special case of the Network model and is not treated separately. Section 5 discusses the power of WCRL and Sec. 6 compares WCRL with other data model independent languages. Finally, in Sec. 7 we offer some concluding remarks.

2. WELL CONNECTED RELATIONS

These have been studied in detail elsewhere.[3] For our purpose here we introduce the definitions of a well connected relation (WCR) and an elementary well connected relation (EWCR).

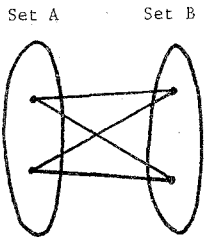


Fig. 1. A well connected relation.

Definition 1. A WCR is a binary relation W on two sets, A and B, such that:

$$(\forall x)(x \in A)(\forall y)(y \in B)(xWy)$$

The sets A and B are called the first and second constituents of the WCR. It is written as W[A; B]. An example of a WCR is shown in Fig. 1.

Definition 2. A WCR is an EWCR if the first constituents ha a single element in it. An example of an EWCR is shown in Fig. 2.

We can extend these ideas to apply to relational databases if we allow sets A and B to be groups of attributes in an attribute relation of an entity. We can extend them to Network and Hierarchical databases, if we allow A and B to be all the attributes in attribute relations of two entities which are linked by an association. Here, A and B in general will refer to groups of attribute names or groups of entity names and not to the actual values. This is unlike the two definitions given above. Sometimes we need to refer to the actual values of the attributes. In that case, A and B will be preceded specifically by the word "Set."

Definition 3. Two WCR's are compatible if the two constituents of one are on the same domains as those of the other.

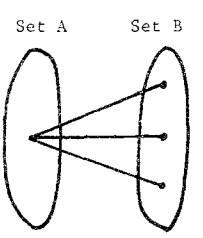


Fig. 2. An elementary well connected relation.

Definition 4. A set of compatible EWCR's in which no two EWCR's have the same value for the first constituent is called a Canonical Partition, represented by l[A; B].

3. THE WCR LANGUAGE (WCRL)

The language WCRL is based on the data structure EWCR. It has efficient set processing because any condition on the first constituent of an EWCR is evaluated only once for all the tuples in the EWCR. The complex conditional expressions in the queries make it possible to use optimization in query processing as suggested by Pecher.[10] WCRL has a low level part and a high level part. The former uses only EWCR's as operands while the latter uses canonical partitions as operands. The operations fall into three classes, namely, Set Reconstitution, Set Join, and Pseudo Set Operations. A Set Join may be a Codd's Join or a Brooming Join while a Pseudo Set Operation may be a Selective Union, a Selective Intersection, or a Selective Difference. Last, it is possible to specify new operations recursively.

3.1. Low Level Language

The various operations are as follows.

3.1.1. Set Reconstitution (σ_{sr})

This operation takes the form

σ_{sr} W[A; B] (First Expression (X); Second Expression (Y)) = l[X; Y]

An expression may be:

1. $X = A_1, A_2, B_1, \ldots$, i.e., a list of attributes or entity names belonging to $(A \cup B)$.
2. $X = \varnothing$, i.e., null.
3. $X = A_1$ (Total-Cost = α(Cost β Qty.)), B_1, where
   a. Total-Cost is a new attribute in X made up of existing attributes.
   b. Cost and Qty. (quantity) in $(A \cup B)$.
   c. α is one of the aggregate functions as follows:

      SUM = summation of all appropriate attribute values in W[A; B] or
      MAX = maximum of all appropriate attribute values in W[A; B] or
      MIN = minimum of all appropriate attribute values in W[A; B] or

CAR = number of distinct values of appropriate attribute in W[A; B] or

AVR = average of all appropriate attribute values in W[A; B].

d. β is arithmetic operations ($+$, $-$, $\div$, $\times$).

4. $X = (A_1 = 1000), (B_1 = AVR(B_2) \times MAX(B_2) \div 1000)$.

The arithmetic operations are evaluated from left to right. Here the attribute A_1 is assigned a single value in all tuples.

3.1.2. Set *Join* (σ_{sj})

This operation takes the form

$$\sigma_{sj}\, W_1[A_1 ; B_1], W_2[A_2 ; B_2], \ldots, W_n[A_n ; B_n](\text{Conditional Expression}) = 1[X; Y]$$

where

$$\left.\begin{array}{l} X = A_1, A_2, \ldots, A_n \\ Y = B_1, B_2, \ldots, B_n \end{array}\right\} \text{Codd's Join } (\sigma_{cj})$$

or

$$\left.\begin{array}{l} X = A_1, B_1, A_2, B_2, \ldots, A_n \\ Y = B_n \end{array}\right\} \text{Brooming Join } (\sigma_{bj})$$

The conditional expression involves attributes from $(A_1 \cup A_2 \cup \cdots A_n \cup B_1 \cup B_2 \cup \cdots \cup B_n)$. An example of a conditional expression is

$$((A_1 = A_2) \wedge (B_1 > 1000) \vee (B_2 \leqslant B_3))$$

The conditional expression consists of a set of conditions connected by Booleans, $\wedge$, $\vee$, $\neg$. Each condition may involve set comparison operations, as follows,

A \ B	CURRENT	SOME	ALL
CURRENT	$>, <, =,$ $\geqslant, \leqslant, \neq$	$>\cdot, <\cdot, =\cdot,$ $\geqslant\cdot, \leqslant\cdot, \neq\cdot$	$>\cdot\cdot, <\cdot\cdot, =\cdot\cdot$ $\geqslant\cdot\cdot, \leqslant\cdot\cdot, \neq\cdot\cdot$
SOME	$\cdot\ , \cdot<, \cdot=$ $\cdot\geqslant, \cdot\leqslant, \cdot\neq$	$\cdot>\cdot, \cdot<\cdot,$ etc	$\cdot>\cdot\cdot, \cdot<\cdot\cdot$ etc.
ALL	$\cdot\cdot>, \cdot\cdot<, \cdot\cdot=$ $\cdot\cdot\geqslant, \cdot\cdot\leqslant, \cdot\cdot\neq$	$\cdot\cdot>\cdot, \cdot\cdot<\cdot$ etc.	$\cdot\cdot<\cdot\cdot, \cdot\cdot<\cdot\cdot$ etc.

Examples

$(A > B)$ the current value of A is greater than the current value of B;
$(A >\cdot B)$ the current value of A is greater than some value of B in the WCR;
$(A \cdot\cdot> \cdot\cdot B)$ all values of A in its WCR are greater than all values of B in its WCR;

and so on.

Further, a condition may involve set containment operations as follows:

$A \supseteq B$ the set of values of A in its WCR contains or equals the set of values of B in its WCR.

Similarly for

$$A \supset B, \quad A \subset B, \quad A \subseteq B, \quad A \not\supset B, \quad A \not\subset \not\subset B, \quad A \not\subseteq B, \quad A \not\supseteq B$$

Also,

$A \equiv B$ the set of values of A in its WCR is equal to the set of values of B in its WCR,

and similarly for $A \not\equiv B$.

3.1.3. Pseudo Set Operations

These are "Selective Union" $(\sigma_{su})\cdot$, "Selective Intersection" (σ_{si}), and "Selective Difference" $(\sigma_{sd})\cdot$.

1. Selective Union (σ_{su}). This operation takes the form

$$\sigma_{su}\, W_1[A_1 ; B_1], W_2[A_2 ; B_2], \ldots, W_n[A_n , B_n] \text{ (Conditional Expression)} = I[A; B]$$

Here all WCR's are compatible and all WCR's which have the same value satisfying the conditional expression, for the first constituent are combined to form a single possible WCR of I[A; B]. From this WCR those tuples which do not satisfy the conditional expression on the second constituent or are duplicates are removed to get a final WCR of I[A; B].

2. Selective Intersection (σ_{si}). This operation takes the form

$$\sigma_{si}\, W_1[A_1 ; B_1], W_2[A_2 ; B_2], \ldots, W_n[A_n , B_n] \text{ (Conditional Expression)} = W[A; B]$$

Here all WCR's are compatible and their common tuples, if any, are selected on the basis of the conditional expression.

3. Selective Difference (σ_{sd}). This operation takes the form

$$\sigma_{sd}\ W_1[A_1\ ;\ B_1],\ W_2[A_2\ ;\ B_2]\ (\text{Conditional Expression}) = W[A;\ B]$$

Here W_1 and W_2 are compatible; their common tuples are removed from W_1 ; and from the remaining tuples of W_1, those satisfying the conditional expression are selected to form W[A; B].

3.2. High Level Language

The various operations are as follows.

3.2.1. Set Reconstitution (σ_{sr})

This operation takes the form

$$\sigma_{sr}\ l_1[A;\ B]\ (\text{First Expression (X)};\ \text{Second Expression (Y)}) = l[X;\ Y]$$

Here the low level operation is performed on each WCR of $l_1[A; B]$. The final result contains no duplicate tuples and is a canonical partition.

3.2.2. Set Join (σ_{sj})

This operation takes the form

$$\sigma_{sj}\ l_1[A_1\ ;\ B_1],\ l_2[A_2\ ;\ B_2],\ ...,\ l_n[A_n\ ;\ B_n]\ (\text{Conditional Expression}) = l[X;\ Y]$$

where

$$\left.\begin{array}{l} X = A_1\ ,\ A_2\ ,...,\ A_n \\ Y = B_1\ ,\ B_2\ ,...,\ B_n \end{array}\right\} \text{Codd's Join } (\sigma_{cj})$$

or

$$\left.\begin{array}{l} X = A_1\ ,\ B_1\ ,\ A_2\ ,\ B_2\ ,...,\ A_n \\ Y = B_n \end{array}\right\} \text{Brooming Join } (\sigma_{bj})$$

Here the low level operation is applied to every n-cardinality set of WCR's taken one each from $l_1[A_1\ ;\ B_1]$, $l_2[A_2\ ;\ B_2]$, ..., $l_n[A_n\ ;\ B_n]$.

3.2.3. Pseudo Set Operations

1. Selective Union (σ_{su}). This operation takes the form

$$\sigma_{su}\ l_1[A_1\ ;\ B_1],\ l_2[A_2\ ;\ B_2],\ ...,\ l_n[A_n\ ;\ B_n]\ (\text{Conditional Expression}) = l[A;\ B]$$

The low level operation is applied to all the WCR's in $l_1[A_1\ ;\ B_1]$, $l_2[A_2\ ;\ B_2]$, etc.

2. Selective Intersection (σ_{si}). This operation takes the form

$$\sigma_{si}\ l_1[A_1\ ;\ B_1],\ l_2[A_2\ ;\ B_2],\ \ldots,\ l_n[A_n\ ;\ B_n]\ \text{(Conditional Expression)} = [A;\ B]$$

The low level operation is applied to every n-cardinality set of WCR's taken one each from $l_1[A_1\ ;\ B_1]$, $l_2[A_2\ ;\ B_2]$, etc.

3. Selective Difference (σ_{sd}). This operation takes the form

$$\sigma_{sd}\ l_1[A_1\ ;\ B_1],\ l_2[A_2\ ;\ B_2]\ \text{(Conditional Expression)} = l[A;\ B]$$

The low level operation is applied to every 2-cardinality set of WCR's taken one each from $l_1[A_1\ ;\ B_1]$ and $l_2[A_2\ ;\ B_2]$.

4. New operations. Several high level operations of WCRL can be combined to specify a new operation. We can do this recursively as well. For example,

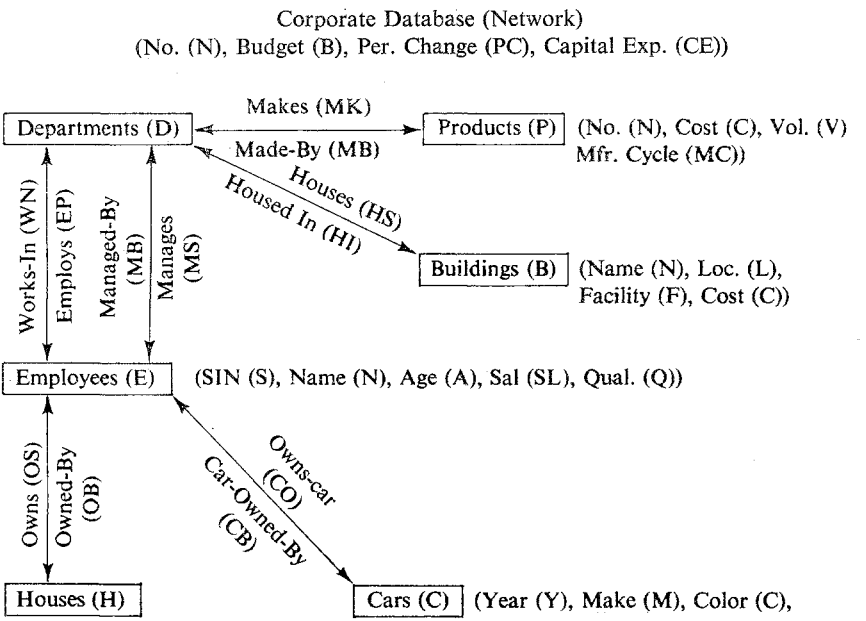
$$\sigma_1\ l[A;\ B] \triangleq \sigma_{sr}\ \{\sigma_1\ \{\sigma_{su}\ l_1[A_1\ ;\ B_1],\ l[A;\ B]\ \text{(Conditional Expression)}\}\}$$
$$\text{(First Expression (X); Second Expression (Y))} \triangleq l_1[X;\ Y]$$

The operation in the innermost set of brackets, { }, is evaluated first. The above definition of σ_1 is recursive. This facility makes WCRL more powerful than relational algebra because it can be used to evaluate least fixed point operations such as transitive closure. Later in this paper we give an example to illustrate this point.

Note. We distinguish among the following conditions that may arise. First, in W[A; B] A may be $\varnothing$, i.e., A is a constituent which has no attributes. In this case all the elements (or tuples) of B form a single WCR. Second, in W[A; B] B may be $\varnothing$, in which case W[A; B] has only one element (or tuple). Third, one of the values of an attribute may be null (#). In this case, the conditional expressions in some of the queries will have an unknown evaluation. We assign an evaluation of "false" to such expressions to avoid an uncontrolled propagation of nulls (#) in the user views. Another complex problem is that of allowing updates to the stored data through user views. A conservative approach, allowing updates only when the user view corresponds to an attribute relationship or an association in the conceptual schema, has been adopted.

4. QUERIES

We show examples of query handling using a Network database (Fig. 3) and a Relational database (Fig. 4). The Hierarchical data model is a special case of the former and is not treated separately. In the following we use



Notes

1. Each entity is linked by two way associations with other entities. These are indicated by the bidirectional arrows.
2. Each one way association represents a collection of WCR's.
3. Each entity has several attributes.
4. Each entity name represents several collections of WCR's depending on which way the attributes are partitioned into the constituents of the WCR's.
5. Every name has a short form representation in brackets which will be used in answering queries. For example an Employee's SIN will be referred to as E.S, his Age as E.A, and so on.
6. The key in each attribute relation is underlined.

Fig. 3. Network database.

abbreviations for attribute, entity, and link names. The full names along with the abbreviations are shown in Figs. 3 and 4.

4.1. Network Database

This database consists of entities (such as Departments, Employees, etc.) which have several attributes. The relationship among the attributes of an

Corporate Data Base (Relational)

Department (D) (Budget (B), Per. Change (PC), Capital Exp. (CE), No. (N), Product No. (PN), Building Name (BN), Employee SIN(ES), Building Loc. (BL), Manager (M))

Employees (E) (Name (NM), Age (A), Sal (SL), Qual. (Q), SIN(S), Department No. (DN), Manages Dept. No. (MN), House No. (HN), House Street (HS), Car Plate No. (CN))

Houses (H) (Price (P), Style (S), No. (N), Street (ST), Owner SIN(OS))
Cars (C) (Year (Y), Make (M), Color (C), Plate No. (PN), Owner SIN(OS))
Products (P) (Cost (C), Vol. (V), Mfr. Cycle (MC), No. (N), Dept. No. (DN))
Buildings (B) (Facility (F), Cost (C), Name (BN), Loc (L), Dept. No. (DN))

Notes

1. Some entries in an attribute may be null (#). For example an employee may not manage any department.
2. The network Corporate Data Base has been translated to the relational model.
3. No attempt is made to see if the relations are in a particular normal form.
4. The data base is taken as such to demonstrate WCRL.

Fig. 4

entity is indicated in Fig. 3 as a relational table. The relationship or association among two entities is always a bidirectional one, represented by a bidirectional arrow in Fig. 3. This form of association representation follows Senko's[11] concept of an "association pair" for fact representation. Each attribute relationship can be viewed as several collections of EWCR's depending on the way the attributes are partitioned among the two constituents of the EWCR's. For example, D[N, B; PC, CE] and D[N; B, PC, CE] are two different collections of EWCR's of the same entity, D. Further each association is a collection of EWCR's in each of the two directions. For example, EP[D; E] and WN[E; D] are collections of EWCR's corresponding to the associations EP and WN as shown in Figs. 3 and 5.

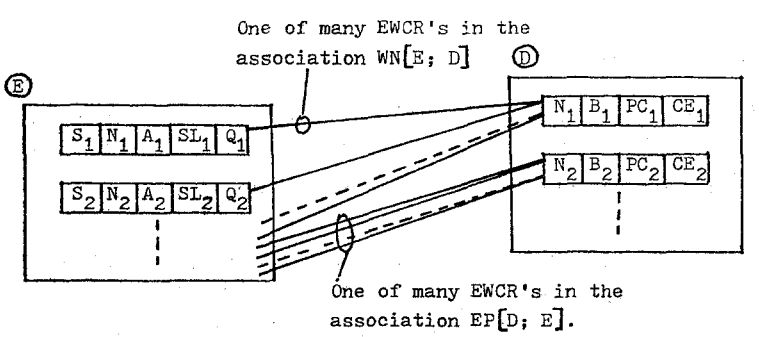


Fig. 5. Representation of an association.

Keeping the above in mind, we present the foljowing query examples. The same queries are answered for the relational database later. We use empty brackets, [ ], for a canonical partition whenever the constituents are obvious from the context.

1. Find all employees, their names, and SIN's, who earn more than their managers.

$$\sigma_{sr}\{\sigma_{cj}\ MB[D; E], EP[D; E]\ ((MB.D = EP.D) \wedge (EP.E.SL > MB.E.SL))\}\ (EP.E.S; EP.E.N) = l_1[E.S; E.N]$$

2. Find the employees, their names, and SIN's, who earn more than their managers, with the addresses of their houses and the plate numbers of their cars.

$$\sigma_{sr}\{\sigma_{cj}\ OS[E; H], CO[E; C], l_1[E.S; E.N]\ ((OS.E.S = CO.E.S) \wedge (OS.E.S = l_1.E.S))\}\ (l_1.E.S; l_1.E.N, OS.H.N, OS.H.ST, CO.C.PN) = l_2[E.S; E.N, H.N, H.ST, C.PN]$$

3. Find the employees, their names, and SIN's, who earn more than their managers, with the product numbers they make and the building names and locations they work in.

$$\sigma_{sr}\{\sigma_{cj}\{\sigma_{sr}\{\sigma_{bj}\ l_1[E.S; E.N], WN[E; D], MK[D; P]\ (l_1.E.S = WN.E.S) \wedge (WN.D = MK.D)\}\ (l_1.E.S, l_1.E.N, WN.D.N; MK.P.N)\}, HI[D; B]\ (WN.D.N = HI.D.N)\}\ (l_1.E.S, l_1.E.N; MK.P.N, HI.B.N, HI.B.L) = l_3[E.S; E.N, P.N, B.N, B.L]$$

4. Find the employees, their names, and SIN's, who earn more than their managers, with the product numbers they make, the building names and locations in which they work, the addresses of their houses, and the plate numbers of their cars.

$$\sigma_{sr}\{\sigma_{cj}\ l_2[\ ], l_3[\ ]\ (l_2.E.S = l_3.E.S)\}\ (l_2.E.S, l_2.E.N, l_2.H.N, l_2.H.ST, l_2.C.PN, l_3.P.N, l_3.B.N, l_3.B.L; \varnothing) = l_4[E.S, E.N, H.N, H.ST, C.PN, P.N, B.N, B.L; \varnothing]$$

4.2. Relational Database

The Network database in Fig. 3 is translated to a Relational database in Fig. 4 and the same queries are answered using the translated database.

1. Find all employees, their names, and SIN's, who earn more than their managers.

$$\sigma_{sr}\ E[\ ]\ (DN; SIN, N, SAL, MN) = l_1[DN; SIN, N, SAL, MN]$$
$$\sigma_{su}\ l_1[\ ]\ (MN \neq \#) = l_2[DN; SIN, N, SAL, MN]$$
$$\sigma_{sr}\{\sigma_{cj}\ l_1[\ ], l_2[\ ]\ (l_1.DN = l_2.DN)\}\ (l_2.SIN \neq l_1.SIN) \wedge (l_1.SAL > l_2.SAL)\}\ (l_1.N, l_1.SIN; \varnothing) = l_3[N, SIN; \varnothing]$$

2. Find the employees, their names, and SIN's, who earn more than their managers, with the addresses of their houses and the plate numbers of their cars.

$$\sigma_{sr}\{\sigma_{cj}\{\sigma_{sr}\ E[\ ](N, SIN; HN, HS, CN)\}, l_3[N, SIN; \varnothing]\ (E.SIN = l_3.SIN)\}\ (l_3.SIN; l_3.N, E.HN, E.HS, E.CN) = l_4[SIN; N, HN, HS, CN]$$

3. Find the employees, their names, and SIN's, who earn more than their managers, with the product numbers they make and the building names and locations they work in.

$$\sigma_{sr}\{\sigma_{cj}\{\sigma_{sr}\ D[\ ](ES; PN, BN, BL)\}, l_3[\ ](D.ES = l_3.SIN)\}\ (l_3.SIN; l_3.N, D.PN, D.BN, D.BL) = l_5[SIN; N, PN, BN, BL]$$

4. Find the employees, their names, and SIN's, who earn more than their managers, with the product numbers they make, the building names and locations in which they work, the address of their houses, and the plate numbers of their cars.

$$\sigma_{sr}\{\sigma_{cj}\ l_4[\ ], l_5[\ ](l_4.SIN = l_5.SIN)\}\ (l_4.SIN, l_4.N, l_4.HN, l_4.HS, l_4.CN, l_5.PN, l_5.BN, l_5.BL; \varnothing) = l_6[SIN, N, HN, HS, CN, PN, BN, BL; \varnothing]$$

5. COMPLETENESS

The language WCRL is more powerful than relational algebra because it can express all the operations of relational algebra and also compute the least fixed point operations such as transitive closure. We give equivalent expressions for relational algebra operations and also show how the transitive closure of a relation may be computed. In the following, the relations on the right-hand side should be viewed as canonical partitions. We use empty brackets whenever their contents are obvious from the context.

1. Projection:

$$R[A] \leftarrow \sigma_{sr}\ R[X\ Y]\ (\varnothing; A)$$

2. Restriction and selection:

$$R[A\sigma B] \leftarrow \sigma_{sr}\{\sigma_{su}\ R[X; Y]\ (A\sigma B)\}\ (\varnothing; X, Y)$$

3. Join:

$$R[A\sigma B]S \leftarrow \sigma_{sr}\{\sigma_{cj}\ R[X; Y], S[P; Q]\ (A\sigma B)\}\ (\varnothing; X, Y, P, Q)$$

4. Division:

$$R[A, B]\,[B \div C]S[C, D] —$$
$$\sigma_{sr}\,\{\sigma_{cj}\,R[A; B], \{\sigma_{sr}\,S[C; D]\,(\varnothing; C, D)\}\,(B \supseteq C)\}\,(\varnothing; A)$$

5. Cartesian product:

$$R \times S — \sigma_{cj}\,R[\varnothing; X], S[\varnothing; Y]$$

6. Union:

$$R \cup S — \sigma_{sr}\,\{\sigma_{su}\,R[X; Y], S[X; Y]\}\,(\varnothing; X, Y)$$

7. Intersection:

$$R \cap S — \sigma_{sr}\,\{\sigma_{si}\,R[X; Y], S[X; Y]\}\,(\varnothing; X, Y)$$

8. Difference:

$$R - S — \sigma_{sr}\,\{\sigma_{sd}\,R[X; Y], S[X; Y]\}\,(\varnothing; X, Y)$$

9. Transitive closure: the transitive closure[15] of a binary relation R, denoted by R^+, is the set of pairs (a, b) such that for some sequence $c_1, c_2, \ldots, c_n$,

a. $c_1 = a$,

b. $c_n = b$,

c. for $i = 1, 2, \ldots, n - 1$ we have (c_i, c_{i+1}) in R.

We define a new operation, σ_1, as follows,

$$\sigma_1\,l[A; B] \triangleq \sigma_1\,\{\sigma_{su}\,\{\sigma_{sr}\,\{\sigma_{bj}\,l[A; B], l[A_1; B_1]\,(B = A_1)\}\,(A; B_1)\}, l[A; B]\} \triangleq l_2[A_2; B_2]$$

Here $l[A; B]$ and $l[A_1; B_1]$ are the same canonical partitions and σ_1 gives as a result $l_2[A_2; B_2]$ which is the transitive closure of $l[A; B]$.

6. COMPARISON OF WCRL, LSL, FQL, AND QUEST

In this section we give examples of queries in the four data model independent languages that may be found in the literature. The intent is to provide the reader with a flavor of these languages. We also give a tabular comparison of the features that may be found in these languages (Table I). The language LSL was published first. We also give a detailed comparison between LSL and WCRL (Table II). We emphasize here that the syntax of WCRL is mathematical and it is our intent to build a user freindly syntax on top of WCRL in the future. This is similar to the example of the SEQUEL and

Table I. Comparison of Data Model Independent Languages

	PROPERTY	FQL	WCRL	LSL	QUEST
1.	Set Comparators	Not Available	Available	Not Available	Partially Available
2.	Recursion	Available	Available	Not Available	Not Available
3.	Arithmetic Operations	Available	Available	Available	Available
4.	Mode	Algebraic	Algebraic but Navigation possible	Navigational	Navigational
5.	Syntax	Mathematical	Mathematical	User Friendly	User Friendly
6.	Function	Data Manipulation	Data Retrieval	Data Definition and Manipulation	Data Manipulation
7.	Power	More Than Relational Algebra	More Than Relational Algebra	Same as Relational Algebra	Same as Relational Algebra
8.	Aggregate Functions	Not Available	Available	Not Available	Available

SQUARE languages,[5] where the latter is mathematical and the former is its user friendly version.

1. With reference to the Network database in Fig. 3, the following data selection first selects those houses which are more expensive than \$100,000, finds their owners and selects those who earn less than \$100,000; then it finds the year of the cars owned by these employees and the budget of the departments they work in. At the end it displays House.Price, Employees.Names, Employees.Sal, Cars.Year, and Departments.Budget.

LSL

SELECT House
WHERE House.Price > 100,000
KEEP House.Price
LINK WITH OWNED-BY *TO* Employees

Table II. A Comparison of LSL with WCRL

	LSL	WCRL
1. Design Philosophy	For an unsophisticated user	For a sophisticated user
2. Design Aim	For relational, network or hierarchical data base	Same
3. Function	Data definition and data manipulation	Data retrieval
4. Design Decisions	i) No projections and selections on booleans of derived relations	i) Allowed
	ii) No joins on derived relations	ii) Allowed
	iii) Joins dependent on associations and limited to natural	iii) Independent and not limited
5. Power	i) Relationally complete	i) More powerful than relational algebra
	ii) Null values are not handled	ii) Possible to handle null values
	iii) Navigational	iii) Algebraic but allows navigation

SELECT Employees
WHERE Employees.Sal $<$ 100,000
KEEP Employees.Name *AND* Employees.Sal
LINK WITH OWNS-CAR *TO* Cars
KEEP Cars.Year
LINK FROM Employees *WITH* WORK-IN *TO* Departments
KEEP Departments.Budget

WCRL

$$\sigma_{sr}\{\sigma_{cj}\ \mathrm{OS[E;\ H],\ CO[E;\ C],\ WN[E;\ D]}\ (\mathrm{OS.E} = \mathrm{CO.E}) \wedge (\mathrm{CO.E} = \mathrm{WN.E}) \wedge (\mathrm{OS.H.P} > 100{,}000) \wedge (\mathrm{OS.E.SL} < 100{,}000)\}\ (\mathrm{OS.E.N,\ OS.H.P,\ OS.E.SL,\ CO.C.Y,\ WN.D.B;}\ \varnothing) = l_1[\mathrm{E.N,\ H.P,\ E.SL,\ C.Y,\ D.B;}\ \varnothing]$$

2. We borrow a relational schema from Ref. 8 in Fig. 6 and answer the following query in QUEST and WCRL. Find, for each department, its employees (their ENO and SKILL) and the children of the employees.

```
DEPT( DNO  TYPE      CITY    )

      5012 ADMIN     DALLAS
      5624 RESEARCH  BOSTON
      5000 RESEARCH  CHICAGO

PERSON( ENO   NAME     ADDRESS   SALARY   GRATUITY   BIRTHDATE )

        50324 JOHNSON  BOSTON    10000      000         1936
        51396 SMITH    DALLAS    15000      250         1933
        26204 JONES    DALLAS    17000      500         1935
        12102 LEE      BOSTON    12000      450         1928
        21180 LEE      DALLAS    15000      450         1941
        41200 CLARK    CHICAGO   16000      000         1947

CHILDREN( ENO   CNAME   BIRTHDATE )

          50324 BILL       1953
          50324 KATE       1956
          51396 JOHN       1962
          21180 BILL       1953

SKILL( ENO   LEVEL   SKNAME     )

       50324   1     PLUMBER
       50324   3     ELECTRICIAN
       26204   3     PLUMBER

EMPLOYEE( ENO   DNO  )    MANAGER( ENO   DNO  )

          50324 5012              51396 5012
          50324 5000              21180 5000
          12102 5012              26204 5624
```

Fig. 6. A Relational schema.

QUEST

```
SELECT DEPT
  SELECT PERSON
  WHERE PERSON.ENO = ANY OF
            SELECT EMPLOYEE
            WHERE EMPLOYEE.DNO = DEPT.DNO
            RETURN(ENO)
            END
    SELECT CHILDREN
    WHERE PERSON.ENO = CHILDREN.ENO
    END
    SELECT SKILL
    WHERE SKILL.ENO = PERSON.ENO
    END
  END
END
```

WCRL

σ_r {σ_{cj} EMPLOYEE[ENO; DNO], SKILL[ENO; LEVEL, SKNAME],
CHILDREN[ENO; CNAME, BIRTHDATE] ((EMPLOYEE.ENO
= SKILL.ENO) ∧ (SKILL.ENO = CHILDREN.ENO))}
(EMPLOYEE.DNO; SKILL, CHILDREN)
= l_1[DNO; SKILL, CHILDREN]

3. We borrow a functional view of a database from Ref. 4 in Fig. 7 and show that WCRL can answer queries in this view as well. However, this process is very much simplified if the functional view of data is converted to an attribute relationship as follows.

R_1[EMPLOYEE.NAME(EN), DEPT.NAME(DN),
MARITALSTATUS(MS), SAL(S)]

The query presented here is, "Find the department names and salaries of all married employees."

FQL

Q1: → *[CHAR, NUM]
= !EMPLOYEE.|MARRIED.*[DEPT.DNAME, SAL];

WCRL

With a functional view of the database;

σ_{sr} {σ_{bj} {σ_{cj} MARRIED [EMPLOYEE; BOOL],
SAL[EMPLOYEE; NUM], DEPT[EMPLOYEE; DEPARTMENT]
(MARRIED.EMPLOYEE = SAL.EMPLOYEE)
∧ (SAL.EMPLOYEE = DEPT.EMPLOYEE)
∧ (MARRIED.BOOL = 1)}, DNAME[DEPARTMENT; CHAR]

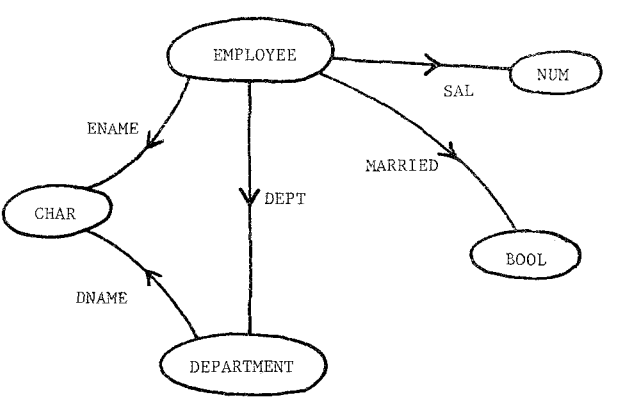


Fig. 7. A functional view of a database.

$$(\text{DEPT.DEPARTMENT} = \text{DNAME.DEPARTMENT})\}$$
$$(\text{DNAME.CNAR, SAL.NUM}; \varnothing)$$
$$= l_1[\text{DNAME, SAL}; \varnothing]$$

With a relational view of the data base,

$$\sigma_{sr}\{\sigma_{su} R_1[\text{EN, DN, MS, S}; \varnothing](\text{MS} = 1)\}(\text{DN, S}; \varnothing)$$
$$= l_1[\text{DNAME, SAL}; \varnothing]$$

7. CONCLUDING REMARKS

In this paper we have presented a data model independent language, WCRL. The language has efficient set processing and also allows more than one comparison in a conditional expression. It applies equally well to the three major data models—Network, Relational, and Hierarchical. It can also handle queries in other models of data. Further work is needed for a user friendly syntax for WCRL. This is under study by the authors.

ACKNOWLEDGMENT

This work was benefitted immensely by several discussions with Dr. K. Sevcik of the University of Toronto.

REFERENCES

1. ANSI/X3/SPARC Study Group on Data Base Management Systems Interim Report, FDT, *ACM SIGMOD* 7 (2) (1975).
2. A. V. Aho and J. D. Ullman, "Universality of data retrieval languages," *Sixth Annual ACM Symposium on Principles of Programming Languages*, San Antonio, Texas (Jan. 1979), pp. 110–120.
3. S. K. Arora and K. C. Smith, "A theory of Well connected relations," *J. Inform. Sci.* (accepted for publication).
4. P. Buneman and R. E. Frankel, "FQL—A Functional Query Language," *ACM SIGMOD*, Boston (1979), pp. 52–58.
5. D. Chamberlin and R. Boyce, "SEQUEL: A Structured English Query Language," *ACM SIGMOD Workshop on Data Description, Access and Control* (May 1974), pp. 249–264.
6. E. F. Codd, "Relational completeness of data base sublanguages," in *Data Base Systems* (R. Rustin, Ed.), Courant Computer Science Symposium (Prentice-Hall Englewood Cliffs, New Jersey, 1972), pp. 65–98.
7. A. L. Furtado and L. Kerschberg, "An algebra of quotient relations," *ACM SIGMOD*, Toronto, (August 1977), pp. 1–9.
8. B. C. Housel, "QUEST: A High-Level Data Manipulation Language for Network, Hierarchical and Relational Databases," IBM Res. Rep. RJ 2588 (33488), 7/25/79 (1979).
9. L. Kerschberg, E. A. Ozkarahan and J. E. S. Pacheco, "A synthetic English query language for a relational associative processor," *Proceedings of the Second International Conference on Software Engineering*, San Francisco (1976), pp. 505–519.

10. R. M. Pecherer, "Efficient evaluation of expressions in a relational algebra," *ACM Pacific 75 Regional Conference* (April 1975), pp. 44–49.
11. M. E. Senko, "The DDL in the context of a multilevel structured description: DIAM II with FORAL," in *Data Base Description*, B. C. M. Douque and G. M. Nijssen, Eds. (North-Holland, Amsterdam, 1975), pp. 239–258.
12. J. M. Smith and P. Y. Chang, "Optimizing the performance of a relational algebra data base interface," *Comm. ACM* 18: 568–579 (October 1975).
13. M. R. Stonebraker and E. Wong, "INGRESS: A relational data base system," *Proceedings of the AFIPS 1975 National Computer Conference*, AFIPS Press, Montvale, New Jersey.
14. D. Tsichritzis, "LSL: A Link and Selector Language," *ACM SIGMOD*, Washington, D. C. (June 1976), pp. 123–134.
15. J. D. Ullman, *Principles of Database Systems*, Computer Science Press, 1980.
16. M. Zloof, "Query by Example," *Proceedings of the AFIPS 1975 National Computer Conference*, AFIPS Press, Montvale, New Jersey, pp. 431–445.