WCRC: AN ANSI SPARC MACHINE ARCHITECTURE FOR DATA BASE MANAGEMENT

SUDHIR K. ARORA and S. R. DUMPALA

Department of Electrical and Computer Engineering
McMaster University
Hamilton, Canada

K. C. SMITH

Department of Electrical Engineering
University of Toronto
Toronto, Canada

ABSTRACT

Several data base machine architectures have been proposed in the past few years. The next generation of these machines must simutaneously support different data models on the same physical data. One approach to this problem, GDBMS, has recently been described in the literature. This paper presents the architecture of another approach: the well-connected relation computer (WCRC). The architectures of these two computers are compared, and a preliminary performance evaluation showing that WCRC requires less storage and has faster query response time is presented.

INTRODUCTION

Data base machine design is a relatively new field. Reasons for the emergence of the field include the following:

1. Users of data base management systems have become more sophisticated. The software to support their demands has grown in complexity. This has created a need for supporting some data base functions at the hardware level.

2. Traditionally, the interface from the central processor to the mass memory of the computer has operated at a very low level. Specialized hardware needs to be designed to free the central processor of this task.

3. LSI technology has reduced the cost of decentralized memory and logic, so pseudoassociative memories with logic-per-track construction are now economically feasible.

Several data base machine architectures have been studied in the literature, including CASSM,[1] RAP,[2] direct,[3] RARES,[4] DBC,[5] RELACS,[6] search processor[7], XDMS,[8] and IFAM.[9] Some of these machines address only specific operations on a data base; some implement only one of three major data models (relational, network, or hierarchical); some implement all three data models, but not simultaneously on the same physical data.

A data base machine architecture that can support different data models simultaneously on the same physical data is needed. Users should be able to view data according to a relational, network, or hierarchical data model (*logical data independence*). Further, changes to the physical data should have little effect on a user's view of the system (*physical data independence*). Such an architecture would conform to the ANSI/X3/SPARC (ANSI 1975) proposals or to the coexistence model,[10] which includes three levels for a data system: external, conceptual, and internal. At the external level, the system would support different data models, depending on user needs. At the conceptual level, a stable common view of data and its semantics would reside. The physical data would be stored at the internal level and could be altered to take advantage of evolving technologies.

To the best of our knowledge, there is only one project in the world addressing this problem: GDBMS.[11] Dogac and Ozkarahan have simulated the ANSI/X3/SPARC proposals or the coexistence model by using the RAP machine as the internal level and software as the conceptual and external levels.

0149-7111/81/0000/0373$00.75 © 1981 IEEE

In this paper, the architecture of a well-connected relation computer (WCRC) is described, and compared with GDBMS.

BACKGROUND

A theory of well-connected relations (WCRs) has been presented elsewhere.[12] For this paper, it is sufficient for the reader to know the following definitions:

Well-Connected Relation (WCR): A binary relation W on two sets A and B such that
$(\forall a)\ (a \in A)\ (\forall b)\ (b \in B)\ (aWb)$.

Sets A and B are called the first and the second *constituent* of WCR.

Elementary Well-Connected Relation (EWCR): WCR in which the first constituent has a single element. The second constituent is called the *image set* of the first constituent.

Trivial Well-Connected Relation (TWCR): WCR in which both constituents have a single element.

A relation $R[A, B]$ can be expressed as:

$$R[A, B] = \sum_{i=1}^{n} R_i[A_i, B_i]$$

$$= R_1[A_1, B_1] \cup R_2[A_2, B_2] \cup \ldots \cup R_n[A_n, B_n]$$

$$= \pi(R) = \text{A partition of } R$$

in which $R_i[A_i, B_i] \cap R_j[A_j, B_j] = \phi$

for $i \neq j$, $1 \leq j \leq n$; and $A = \bigcup_{i=1}^{n} A_i$ and $B = \bigcup_{i=1}^{n} B_i$.

Canonical Partition: A partition of a binary relation $R[A, B]$ if

$$R[A, B] = \sum_{i=1}^{n} W_i[A_i; B_i]$$

in which

1. $W_i\ [A_i; B_i]$ is WCR for $1 \leq i \leq n$;
2. A_i is a set with a single element for $1 \leq i \leq n$;
3. $A_i \neq A_j$ for $i \neq j$ and $1 \leq i, j \leq n$.

Pseudocanonical Pattern (PCP): A partition of a binary relation $R[A, B]$ if

$$R[A, B] = \sum_{i=1}^{n} W_i[A_i; B_i]$$

in which

1. $W_i\ [A_i; B_i]$ is WCR for $1 \leq i \leq n$;
2. A_i is a set with a single element for $1 \leq i \leq n$.

Later in this paper, PCPs are used for storing data in WCRC. A language based on WCRs has been proposed for data base systems.[13] The language is data model independent and can apply equally well to network, relational, or hierarchical data models. The need for data model independent languages has become apparent in recent years. They can be used for the conceptual level of an ANSI SPARC architecture, for communication in a distributed data base system, for data base restructuring, and so on. To the best of our knowledge, the data model independent languages in the literature are FQL,[14] QUEST,[15] LSL,[16] and WCRL. Here, WCRL is used as the language for the conceptual level of WCRC.

WCRC ARCHITECTURE

The WCRC architecture for a data base machine is one hardware version of the ANSI/X3/SPARC architecture or the coexistence model. It has an external level, a conceptual level, and an internal level. It can handle queries from several users simultaneously. It can support the three major data models–network, relational, and hierarchical–simultaneously on the same data at the internal level and may be used as a back-end to a host computer or as an independent data base computer for nonnumeric processing.

The internal level of the machine stores data as WCR partitions called pseudocanonical partitions (PCPs). This is a radical departure from the other approaches in the literature. Other researchers have pursued the policy that at the internal level, related data must be stored contiguously for fast and efficient retrieval. In GDBMS,

for example, the internal level stores data as n-ary relations. It is our contention that in ANSI/X3/SPARC architecture, data must be able to be organized in any way different users may want to see it (flexibility). This will aid in using data efficiently to suit different data model needs at the external level. Hence, the storage unit of data must be simple and small.

In WCRC, the data is stored as WCRs. The well-connected relations themselves are organized as PCPs. This organization aids in response time independency from query statistics. Further, there is greater flexibility in query optimization and maintenance of semantic integrity. Finally, because of their simple structure, PCPs are easy to manipulate. The storage structure is designed to make retrieval of data equally fast for either constituent of WCR.

The conceptual level of the machine supports the conceptual model and the user subschemas in the three major data models. The conceptual-level model is the entity-relationship (E-R) model.[17] The language at the conceptual level is WCRL. The conceptual level also has facilities for query optimization and translation from WCRL to machine language (WCRML) and a data dictionary.

The external level of the machine supports several users working in different data models and translates user languages to WCRL. Each user language has its own DDL. The data base administration (DBA) has a special high-level version of WCRL.

The overall archticture of WRC is shown in Figure 1.

Internal Level

The internal level comprises a number of query processors (QPs) and an array of cell processors (CPs). Each query processor is a master processor responsible for executing one query in machine language, using the cells. The cells are logic-per-track devices. Data is stored on the cell tracks as PCPs. All cells are independent of each other. There is no direct communciation among them; they can communicate only through the query processor that controls them. Data on the cells can be read by any number of query processors, but only one query process can write onto the track at a time. The query processor selects the cells required by the query it is handling and makes them "slaves." They are released once the query is processed. The query processor coordinates its slaves and also computes the overall set results. Several queries can be handled at the same time, because the query processor can work in parallel. This is a multiple instruction-multiple data stream (MIMD) organization.

The memory at the internal level is divided into two parts: (1) permanent memory, which stores the data base, and (2) temporary memory, which stores any intermediate results of a query. Data is organized on the tracks as PCPs. A pseudocanonical partition is a canonical partition in which several WCRs may have the same element as the first constituent. In a canonical partition, WCRs may be arbitrarily large (number of tuples). In PCP, WCRs can be made equal by breaking larger WCRs into several smaller ones, and the hardware size of each WCR can be standardized. This may be done at the system level, the PCP level, or the track level. Because the size of each WCR is fixed, a "fixed" position can be assigned to it on the track. This reduces data movement on the tracks and eliminates "garbage collection" and end markers.

A pseudocanonical partition has two constituents, and logically, either can be the first constituent. This gives two possible PCPs for the same binary data–one forward and one reverse PCP. Data is organized on the tracks of memory in such a way that queries on either PCP can be easily answered, and there is data duplication.

The track format is shown in Figure 2. Each track is divided into two halves–one for each constituent of PCP. The beginning of each half is indicated by storing a header containing the number of WCRs in that half, the attribute names of that constituent, and some flags common to all WCRs in that half of the track. Each standard-sized WCR comprises a set of mark bits, an address field containing the relative position of WCR, a value field containing a value of the first constituent, and a fixed number of pointers. The pointers contain addresses of values in the second constituent of WCR in the other half of the track. Thus, all pointers in one half of the track point to addresses in the other half of the same track, and excessive pointer chasing is avoided. Figure 3 shows an example of a binary relation stored as PCP on a track.

Conceptual Level

The entity-relationship (E-R) model has been chosen to represent the conceptual schema. This model is most suitable for our needs because:

1. The model can easily and naturally be derived from binary PCPs at the internal level.

2. The model incorporates semantic information about data.

3. It avoids many problems associated with normalization. It always achieves at least 3NF.

4. Derivation of other data models is easily done.

5. It allows changes in the schema, providing facility to handle data base growth.

There are four basic objects in this model: *entities*, *relationships*, *attributes*, and *value sets*:

- Entities are items that can be distinctly identified during the design of a data base; for example, employee, company, house. A set of entities with certain common properties constitute an entity set.

- Relationships are associations among entities; for example, work is a relationship between employee and company. Generally, it can be a one-to-one a one-to$N$, or an N-to-M type of association. A set of same-type relationships among the entities is called a relationship set.

- Value sets describe information about an entity or a relationship; for example, number of years is a value set that describes age for the entitiy, employee.

- Attributes are functions that map from entities or relationship sets into value sets or cross products of value sets; for example, age maps the entity set, employee, into the value set, number of years.

In addition, the function performed by an entitiy in a relationship can be specified by *role*; for example, the relationship among employees may have the roles manager and employee.

An entity is identified by its primary key; a relationship, by a combination of primary keys of the entities involved. Sometimes, if an entity cannot be uniquely identified by its own attributes, a relationship is used for identification. Such a *weak entity* is identified by one of its attributes and the primary key of the entity supporting it through the relationship. Thus, entities and relationships can be organized as relations, the attribute names and role names forming the intension and the cross product of value sets forming the tuples of the extension. Certain integrity and consistency constraints may be specified on the value sets (such as allowable values and permitted values) and on relationships such as a one-to-N relationship.

The model also allows for changes caused by data base growth. Entities and relationships constitute the upper domain, and attributes and value sets constitute the lower domain. The following operations are used to effect the changes in both the domains: SPLIT, MERGE, ADD, and DELETE. Finally, SHIFT is used as an operation between the domains; for example, to change a value set into an entity. Figure 4 shows an example of a conceptual schema expressed as an E-R diagram.

A relational model can easily be derived from the E-R model. The entity and relationship relations correspond to 3NF relations in a relational model. The semantic information of functional dependencies is maintained as functional dependencies among the entity and relationship relations.

The network model can be derived from the E-R model by treating entity relations as record types and relationship relations as links. The data structure diagram representation of the network model can be obtained from the E-R model as follows: All one-to-N binary relationships become unidirectional arrows in the diagram (Figure 4); for N-to-M relationships, a new record type is created, and pointed arrows are drawn from the entities involved. The same holds for k-ary relationships ($k > 2$). The hierarchical model is a special case of the network model and can be derived on similar grounds.

The conceptual level (Figure 1) comprises eight functional blocks: the controller, the query analyzer (QA), the query translater (QT), three user schema descriptors (USDs), a conceptual schema descriptor (CSD), and a buffer memory (BM). The conceptual processor maintains the information about the conceptual model and about the storage structure at the internal level. The query analyzer breaks up the queries into subqueries on the binary psuedocanonical partitions (PCPs) stored at the internal level. The user schema descriptors and CSD contain information about the schema definition, operations on the schema, and constraints. This information is used by the query analyzer during query analysis. The query translator translates analyzed queries in WCRL into machine language primitives that can directly be executed by the internal level and stores them in buffer memory.

External Level

The external level performs the following functions:

1. Queues the jobs (or queries)
2. Priority encodes
3. Translates user languages into conceptual-level language

The level has three memory areas: the user work area (UWA), the interface buffer area (IBA), and the processor memory area (PMA). A portion of UWA is allotted to each user as his or her work space. The user can define his (her) own view of data base and specify the constraints on it in this area. The area also acts as an input/output (I/O) buffer between the system and the user.

The interface buffer area is organized as a first-in, first out (FIFO) memory in which queries are stored after they are translated from user languages into WCRL. Translation is handled by three translators stored as software modules in the processor memory. Jobs may also be ranked on a preassigned job priority encoding scheme.

Queries originate at the user end in three languages: LSL, SEQUEL,[18] and IMS.[19] These are translated into WCRL, priority encoded, and put on a queue into IBA by the external processor. The system also supports a data base administrator (DBA), who works directly on the conceptual level, but reaches it through the external level like any other user. The administrator is provided with the facilities of data definition, storage definition, and data manipulation.

Once a query is checked for syntax, translated, and put on the job queue, the conceptual level is activated. It takes one query at a time from IBA and performs query analysis on it. It breaks up the query on logical WCRs into subqueries on physically stored WCRs. It checks whether or not logical WCRs can be constructed from the stored ones and also subjects them to security, integrity, and consistency constraints; any violation of these constraints would terminate the query at this stage, and an error message would be channeled to the user through the external level. The successful queries are then translated by the query translator into machine language and submitted to the internal level through buffer memory. The query translator also provides the information about the required cells along with the machine primitives.

The query processors take the queries from the buffer memory and select the required tracks by a polling scheme. The machine language primitives are executed on the slave cell processors. For the retrieval queries, the data is sent to the external level directly through an I/O read mechanism. For the update queries, the success/or failure ratings are communicated to the user.

PERFORMANCE

A preliminary performance evaluation of the system based on the PCP data structure was done. The storage requirements and computation time for some typical queries when data was stored as PCPs was compared with the same when data was stored as large n-ary relations, as in GDBMS.

Storage Requirements

The following conditions were assigned in comparing the PCP storage structure with the large relational storage structure:

1. In both cases, each value in non-key domains requires 8 bytes of storage.
2. The keys are stored in coded format in both storage schemes.
3. The maximum track size in PCPs is limited to 1.0×10^6 bits.

Based on assumption 3, the maximum number of tuples N that can be accommodated on one PCP track is found as follows: Let m be the average size of WCR in PCP, P_1 be the storage for one pointer in bits, and P_2 be the storage needed for storing one key value in coded format (bits).

Total storage S of a PCP track is 1.0×10^6 bits.

Now, S equals storage for first constituent values (key) plus storage for second constituent values plus storage for the pointers; that is,

$$S = NP_2 + 64(N/m) + 2Np_1 \text{ bits.}$$

The number of second constituent values equals N/m.

The number of total pointers equals $2N$; therefore,

$$N(p_2 + 2p_1 + 64/m) = 10^6. \quad (1)$$

But, say $p_1 = p_2 = \lceil \log_2 N \rceil = p$; therefore,

$$p = \lceil \log_2 N \rceil. \quad (2)$$

(Note that pointers are needed to point to only half of the track.)

From equations (1) and (2):

$$3N\lceil \log_2 N \rceil + 64N/m = 10^6.$$

For a typical value of m, N can now be obtained. Let m equal 200; then

$$3N\lceil \log_2 N \rceil + 64N/200 = 10^6$$

$$N[3\lceil \log_2 N \rceil + 64/200] = 10^6$$

and N $\simeq$ 22,000.

For m = 100, $N \simeq$ 22,000; for m = 10, $N \simeq$ 20,000.

Hence, the track can hold up to 22,000 tuples, and this is largely independent of the size of WCR (m).

Now, let S_1 be the storage required for one relation when stored as a large structure, S_2 be the storage required when same data is stored as PCPs, and M be the number of attributes in the relation. Then,

S_1 = Nx {storage for key + storage for (M - 1) domain values}
= $N[\lceil \log L_2 N \rceil + (M - 1) \times 64]$
= $N\lceil \log_2 N \rceil + 64(M - 1)N$; (3)

and S_2 = Number of PCPs {storage for one PCP}
= (M - 1) {storage for values + storage for pointers}
= (M - 1) {(storage for one key)N + (storage for one non-key domain value)N/m + (storage per pointer)$2N$}
= (M = 1) $[N\lceil \log_2 N \rceil + 64N/m + 2N \log_2 N]$
= (M - 1)$[3N\lceil \log_2 N \rceil + 64N/m]$.

The variations in S_1 and S_2 with reference to N and M are shown in Figures 5, 6, and 7:

- Figure 5 shows that the storage requirement is less for the PCP structure.
- Figure 6 shows the variations of S_1 and S_2 with reference to M when N is kept constant. Again, the PCP storage scheme takes less storage.
- Figure 7 shows how storage requirements S_1 and S_2 vary as the storage size of non-key domain values is changed. It is clear from the graph that for storage sizes greater than 40 bits, it is economical to use the PCP storage scheme.

Queries and Their Computation Times

If data is stored as large relational structures and queries are addressed on only small parts of these, the structures have to be broken up and joined again. If small elementary structures like PCPs are stored, data required by queries can be generated from these, saving computation time. For a class of queries involving a single n-ary relation, the GDBMS may be faster.

Because the PCP tracks are independent of each other, they are processed in parallel. The final results are constructed or assembled from these individual tracks. These aspects are demonstrated in the following example queries.

Consider the data base example in Figure 8. (The storage structures for n-ary relations and PCPs are shown in Figures 9 and 10).

Q1

Find the number of the employee whose salary is greater than $10,000.

Call the storage format in Figure 9 type 1 and the storage format in Figure 10 type 2. Note that for simplicity, the I/O time is not considered in calculating the time taken for the query. Only the marking time is considered.

The Q1 time taken if type 1 format is used equals $t_R + t_p$ in which t_R equals time for one revolution of the track of type 1, which is enough for selection, and t_p equals time taken for projection.

If N_D is the number of distinct elements in a domain over which projection is taken, t_p is given approximately by the expression

$$t_p = N_D(1 + fr)t_R$$

in which fr is a fraction of a revolution. It takes $N_D(1 + fr)$ revolutions to eliminate duplicates; therefore,

$$t_d \approx N_D t_R \{ \text{ neglecting fr} \},$$

and the time taken for Q1 in type 1 format equals approximately $(1 + N_D)t_r$. (5)

The computation time when type 2 format is used is just the time taken to select those tuples that satisfy the condition in Q1. There is not projection involved, because PCP2 contains all items required by the query, without duplication; therefore
time taken for Q1 in type 2 format = t_R. (6)

From equations (5) and (6) it is clear that in type 2 format, the computation time is much less than in type 1.

Q2

Retrieve the numbers and ages of those employees who earn more than $10,000.

This query is similar to Q1, except that for those employees who satisfy the selection clause, age also has to be retrieved. In the first approach, this can be accomplished by first selecting the tuples and then projecting them over attributes ENO and AGE in the relation EMP. Selection would require one revolution, and projection would take approximately N_D revolutions; therefore, the time taken in type 1 format equals $(1 + N_D)t_R$. (7)

In the second approach, the tuples in PCP2 are marked according to the selection clause. The ENO attribute of these selected tuples would be used to cross mark the tuples that agreed on ENO with those in PCP3 (ENO; AGE). Selection on PCP2 would require one revolution and cross marking would take N_D/K revolutions to get the source values, K at a time, and another revolution to finish marking in parallel those target tuples that agree with these K source values on a certain domain; therefore, the time taken in type 2 format equals $(1 + 1 + N_D/K)t_R$. (8)

From equations (7) and (8), it is easy to see that the second approach results in less computation time.

Q3

Find all department numbers and names of employees who work on the second floor of the SUNTOWER building.

In the first approach, the sequence of operations would be:

1. Select tuples in relation DEPT that satisfy ADDR=SUNTOWER and FLOOR=2.
2. Cross mark tuples in EMP=DEPT relation that agree on DNO with the selected tuples.
3. Cross mark tuples in EMP using the marked tuples of EMP=DEPT that agree on ENO.
4. Project NAME from EMP, and DNO from DEPT.

This time taken in the first approach would equal

$$t_R + 2(n/K)t_R + 2(p/K)t_R + (nt_R + mt_R). \quad (9)$$

in which n equals the number of qualified tuples in DEPT relation, p equals the number of qualified tuples in EMP-DEPT relation, and m equals the number of qualified tuples in EMP relation.

In the second approach, the following steps would be:

1. Select tuples in PCP6 (DNO; FLOOR) for which FLOOR=2.
2. Select tuples in PCP5 (DNO; ADDR) for which ADDR=SUNTOWER.
3. Cross mark those marked tuples in PCP5 that agree on DNO with the marked tuples in PCP6.
4. Cross mark tuples in PCP8 (DNO; ENO) that agree with DNO of marked tuples in step 3.
5. Mark those tuples in PCP1 (ENO; NAME) that agree with ENO of marked tuples in PCP8.

The time taken in the second approach 2 would equal

$$t_R + (1 + n_1/K)t_R + (1 + n_2/K)t_R + (1 + n_3/K)t_R. \quad (10)$$

Note that it takes only one revolution for steps 1 and 2, because they can be run in parallel. Here, n_1 equals the number of tuples qualified in PCP6, and n_2 equals the number of tuples qualified in PCP5. Also, $n_1 \leq n$ (n may

have some duplicates), n_3 equals the number of tuples qualified in PCP8, and $n_3 \leq p$ (p may have some duplicates). Therefore, equation (9) is

$$t_1 = t_R(1 + 2n/K + 2p/K + n + m),$$

and equation (10) is

$$t_2 = t_R (4 + n_1/K + n_2/K + n_3/K).$$

It is easy to see that $t_2 < t_1$, because

$$n_1 \leq n,\ n_3 \leq p, \text{ and } n_2 \leq (n + m).$$

COMPARISON OF GDBMS AND WCRC

Following are similarities and differences noted between the GDBMS and WCRC architectures:

1. GDBMS uses RAP as its internal level. Data storage is n-ary relations, which limits efficient use of data in data models other than relational and could restrict query optimization. WCRC uses binary PCPs based on WCRs as the storage structure, which avoids these problems. It further avoids any bias between the forward and the reverse PCPs by a data storage scheme that splits the track into two equal parts.

2. GDBMS performance depends on query statistics; for example, it would be faster for queries in relational data bases. WCRC is independent of query statistics because of its binary PCPs.

3. In GDBMS, the user views are translated directly into the internal level RAP primitives without going through the conceptual level. The limits physical data independence severely. In WCRC, the conceptual level separates the external and the internal level, which results in better physical data independence.

4. In GDBMS, DBA has no DML at the conceptual level, unlike WCRC.

5. In GDBMS, least fixed-point operators and null values are not handled. These facilities are provided in WCRC.

6. GDBMS is an SIMD scheme; WCRC is an MIMD scheme.

7. In GDBMS, the space requirement for n-ary relations is greater than that for the PCP data structure in WCRC. Also, the response time for queries improves in the case of PCP data structure.

CURRENT RESEARCH

Several aspects fo WCRC that are being investigated include:

1. Translation of user languages in relational, network, and hierarchical data bases to the conceptual language WCRL

2. A high-level DBA interface for WCRL

3. Translation algorithms to ensure integrity of user-defined schemas to the entity-relationship model at the conceptual level

4. Translation of WCRL to WCRML (WCR machine language), which manipulates PCPs at the internal level

SUMMARY

The well-connected relation computer (WCRC) is a data base machine that can support different data models simultaneously on the same physical data. The architecture of WCRC has three levels: external, conceptual, and internal. At the external level, the three major data models–network, relational, and hierarchical–are supported, and the data base administrator is provided with a special interface. At the conceptual level, a stable view of the data base can be implemented in the entity-relationship model, and data model independent language based on WCRs (WCRL[13]) is used.

At the internal level, the physical data is stored in logic-per-track pseudoassociative memory. The approach taken by other researchers in this area has been to store related data contiguously as n-ary relations (for example). In WCRC, data is stored as partitions of binary relations, or pseudocanonical partitions (PCPs). Contiguity of data is replaced by pointers. Excessive pointer chasing is avoided, because all pointers in one half of the track point to data on the other half of the track, and vice versa. A preliminary performance evaluation of PCPs compared with n-ary relations showed that PCPs require less storage and

delivered faster computation time for typical user queries.

To the best of our knowledge, there is only one other project in the world, GDBMS,[11] studying simultaneous data base support.

ACKNOWLEDGMENT

This research was supported in part by Science and Engineering Research grant 214-7248.

REFERENCES

[1]G. P. Copeland, G. J. Lipovski, and S. Y. W. Su. The architecture of CASSM: A cellular system for non-numeric processing, *First Annual Symposium on Computer Architecture*, (1973).

[2]E. A. Ozkarahan, S. A. Schuster, and K. C. Smith. RAP–An associative processor for data base management, *National Computer Conference* (1975), 379-387.

[3]D. J. DeWitt. DIRECT–A multiprocessor organization for supporting relational data base management systems, *Proceedings of the 5th Annual Symposium on Computer Architecture (1978), 182-189.*

[4]C. S. Lin, D. C. P. Smith, and J. M. Smith. The design of a rotating associative memory for relational data base applications, *ACM TODS*, 1 (March 1976), 53-65.

[5]J. Banerjee, D. K. Hsiao, and K. Kannan. DBC–A database computer for very large databases, *IEEE Transactions on Computers*, C-28 (June 1979), 414-429.

[6]E. J. Oliver. RELACS, an associative computer architecture to support a relational data model (Ph.D. thesis), Syracuse University, 1979.

[7]H. O. Leilich, G. Stiege, H. Zeidler. A search processor for data base management systems, *Proceedings of the Fourth VLDB* (Sept. 1978), 280-287.

[8]R. H. Canaday et al. A back-end computer for data base management, *CACM* 17 (Oct. 1974), 575-582.

[9]C. R. DeFiore and P. B. Berra. A data management system utilizing an associative memory, *Proceedings of the ACM National Computer Conference* (1973), 181-185.

[10]G. M. Nijassen. A gross architecture for the next generation database management systems, *Modeling in Data Base Management Systems* (New York: North-Holland, 1976), 1-24.

[11]A. Dogac and E. A. Ozkarahan. A generalized DBMS implementation on a database machine, *ACM SIGMOD* (May 1980), 133-143.

[12]S. K. Arora and K. C. Smith. A theory of well-connected relations, *Journal of Information Sciences*, 19 (1979), 97-134.

[13]S. K. Arora and K. C. Smith. WCRL: A data model independent language for data base systems, *International Journal of Computer and Information Sciences*, (1980).

[14]P. Buneman and R. E. Frankel. FQL–A functional query language, *ACM SIGMOD*, (1979), 52-58.

[15]B. C. Housel. QUEST: A high level data manipulation language for network, hierarchical and relational databases (IBM research report BJ2588[33488]), 1979.

[16]D. Tsichritzis. LSL: A link and selector language, *ACM-SIGMOD* (June 1978), 123-134.

[17]P. P. S. Chen. The entity relationship model–Towards a unified view of data, *ACM Transactions on Database Systems*, 1 (March 1976), 9-36.

[18]D. Chamberlin and R. Boyce. SEQUEL: A structured English query language, *ACM-SIGMOD Workshop on Data Description, Access and Control* (May 1979), 249-264.

[19]C. M. Date. *An Introduction to Database Systems*, 2nd ed. (Reading, Mass: Addision-Wesley, 1977).

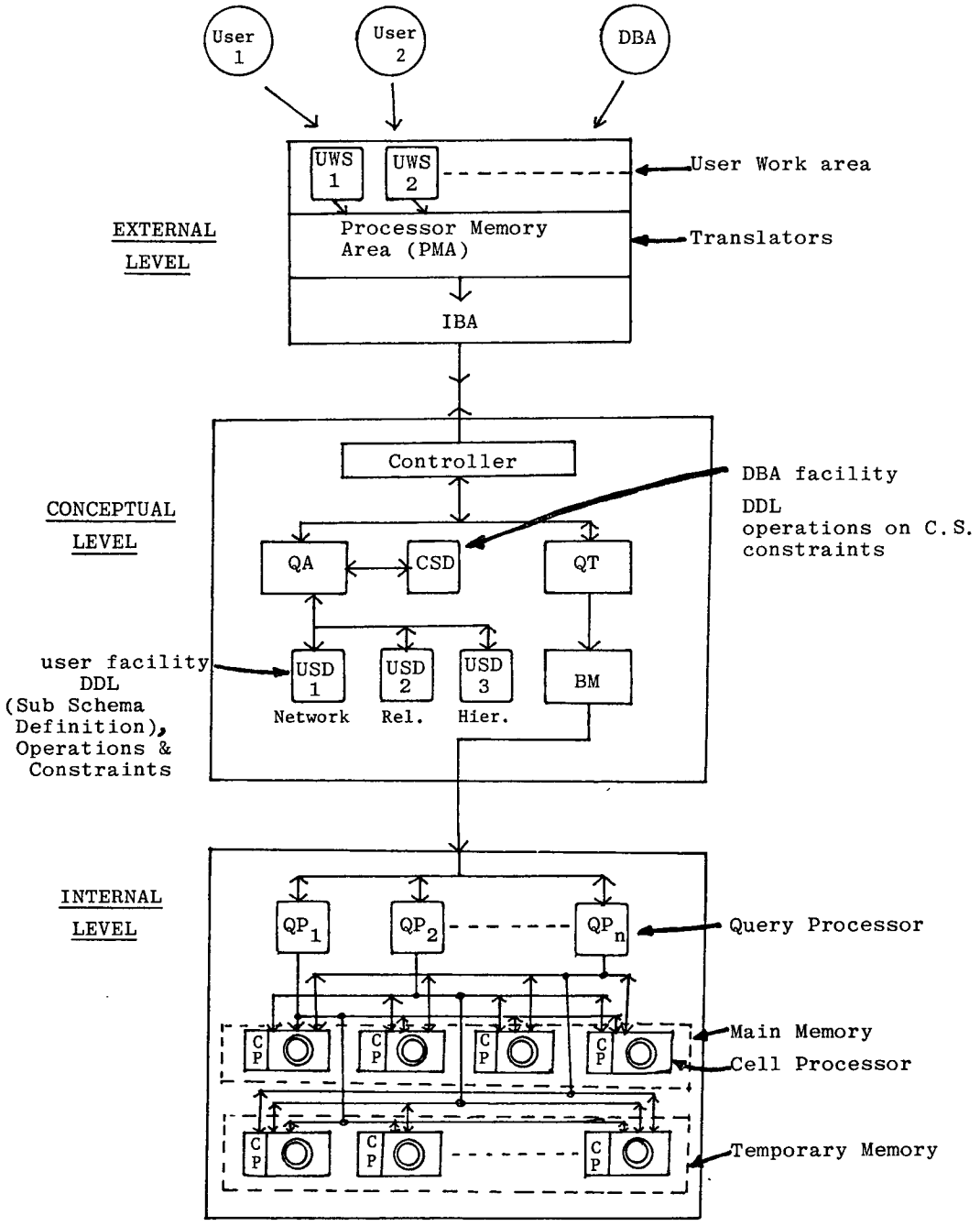

Figure 1. Overall Architecture of WCRL

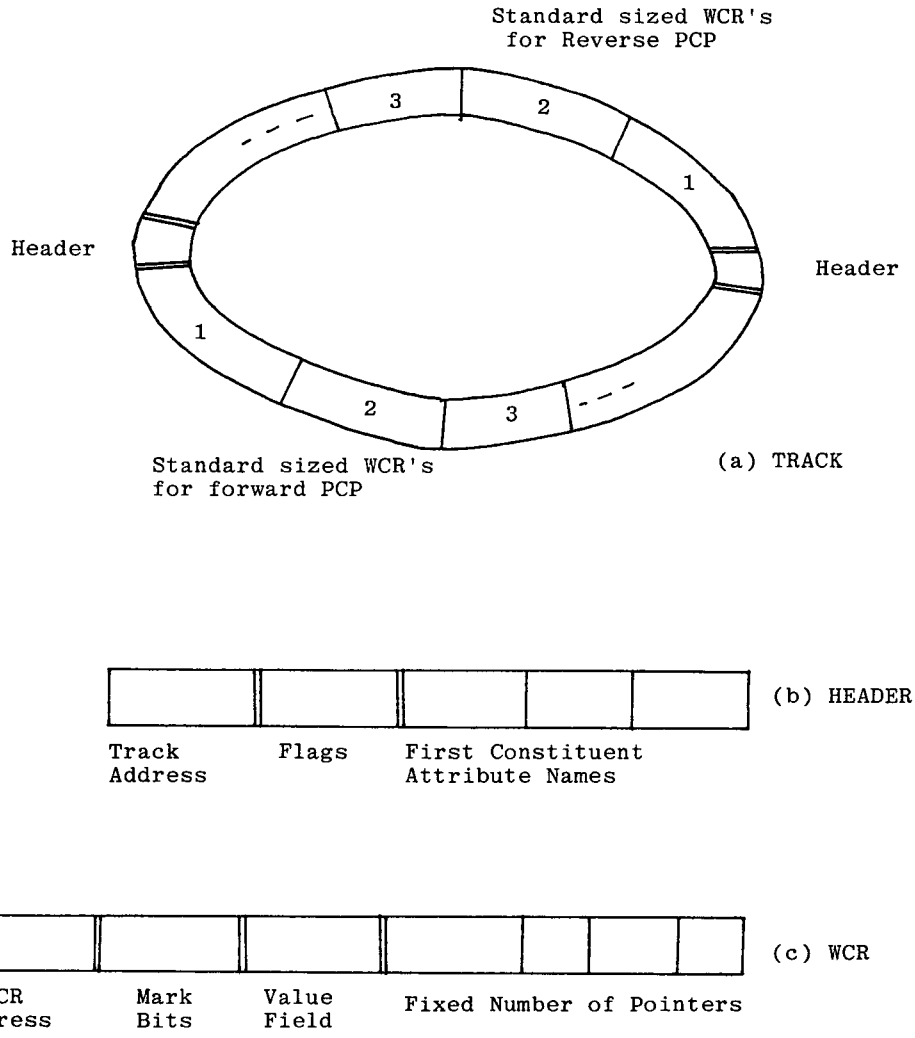

Figure 2. Track Format

a_1	b_1
a_1	b_2
a_1	b_3
a_2	b_2
a_2	b_1
a_3	b_1
a_3	b_2

(a) Binary Relation

(1)	(, , ,)	(a_1)	(1, 2, 3)
(2)	(, , ,)	(a_2)	(1, 2, ϕ)
(3)	(, , ,)	(a_3)	(1, 2, ϕ)

(b) One half of track

(1)	(, , ,)	(b_1)	(1, 2, 3)
(2)	(, , ,)	(b_2)	(1, 2, 3)
(3)	(, , ,)	(b_3)	(1, ϕ, ϕ)

(c) Other half of track

Figure 3. PCPs on a Track

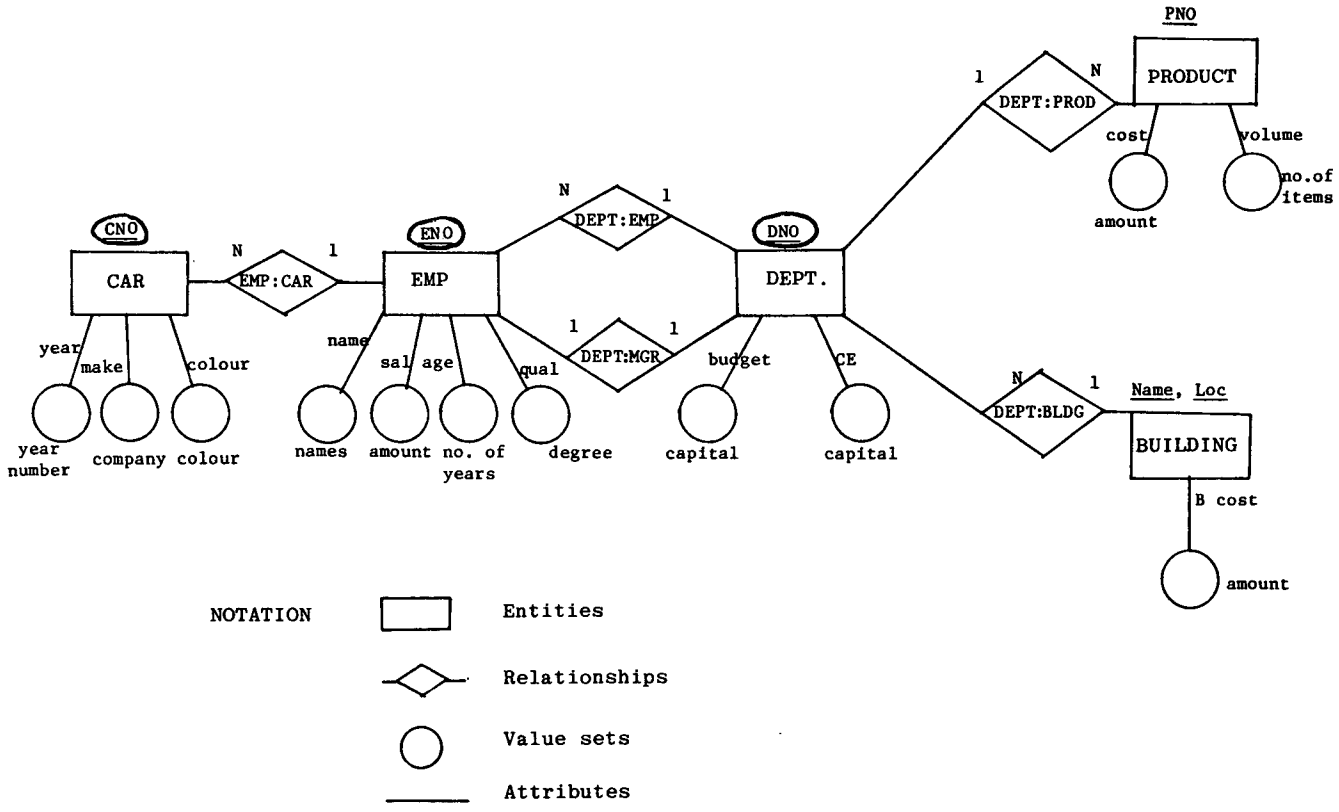

Figure 4. E-R Diagram of Corporate Data Base

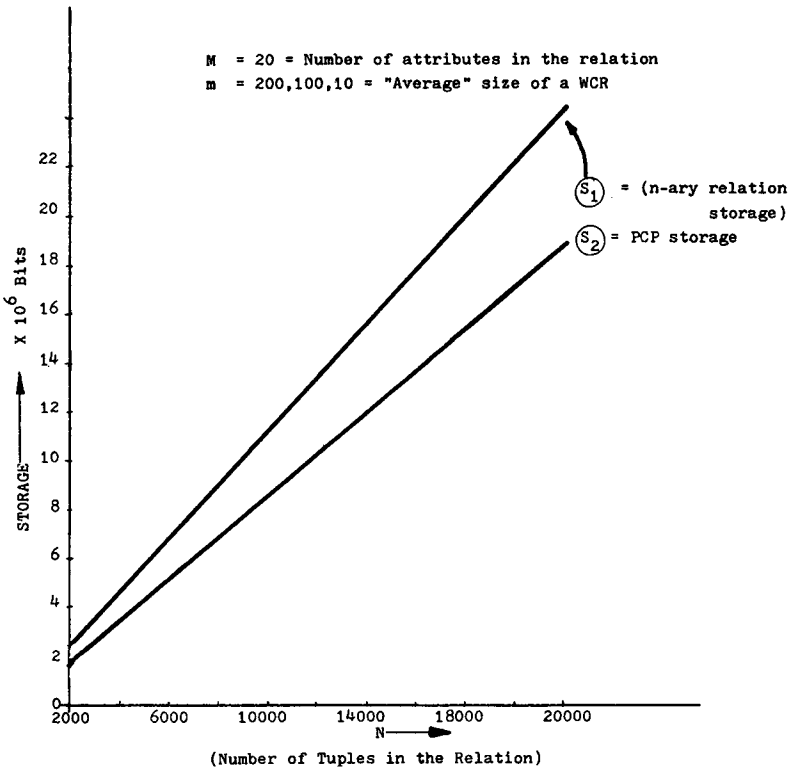

Figure 5. Storage vs Tuples in Relation

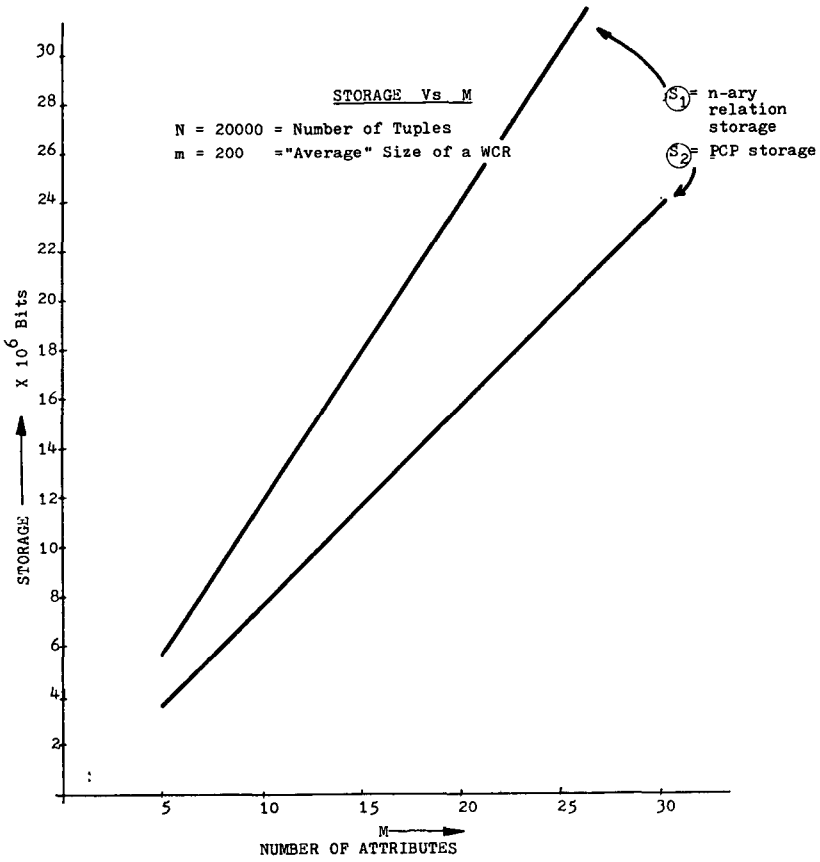

Figure 6. Storage vs Number of Attributes

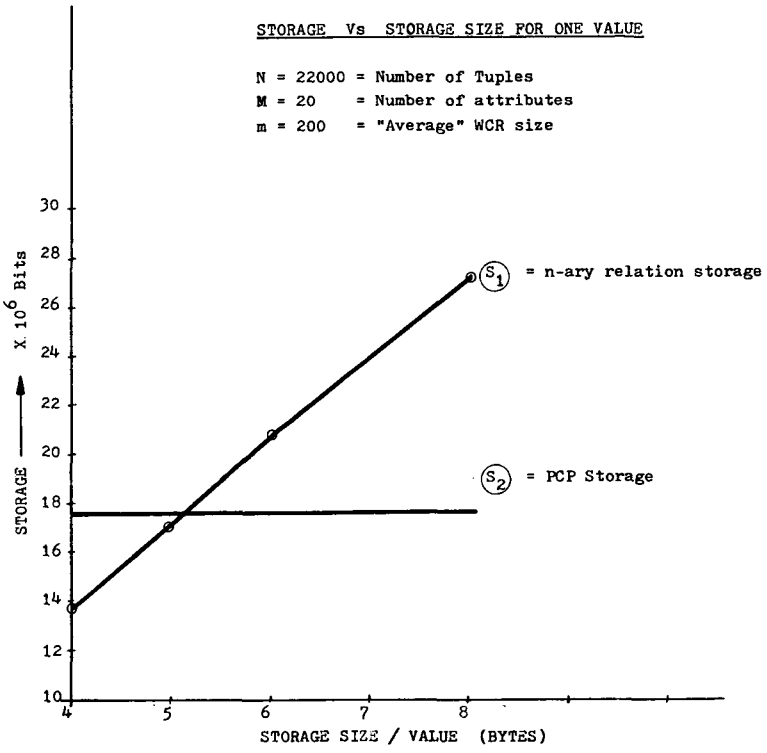

Figure 7. Storage vs Storage Size for One Value

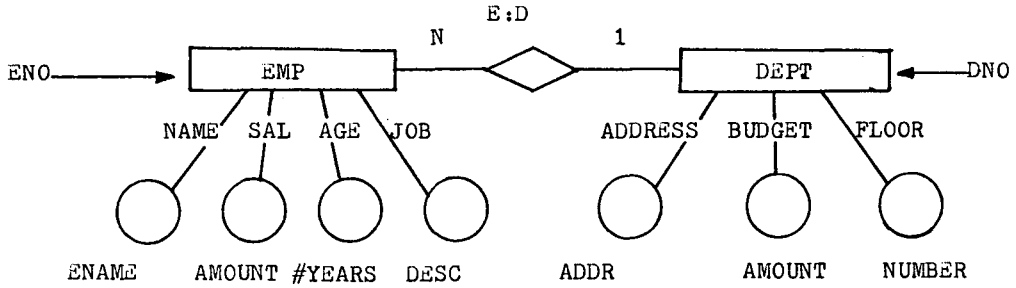

Figure 8. E-R Diagram

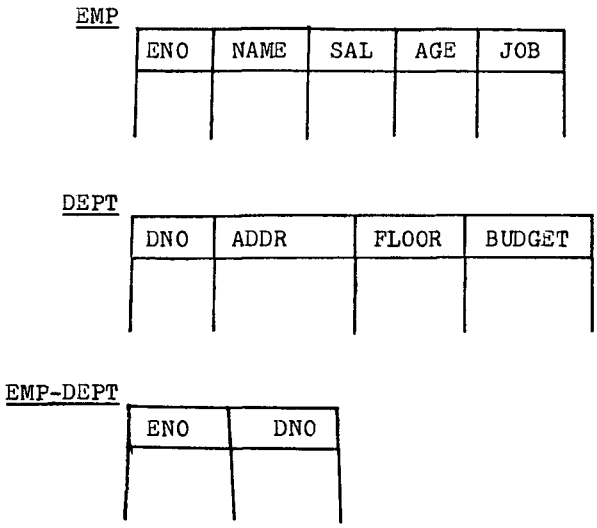

Figure 9. Storing Data as Relations

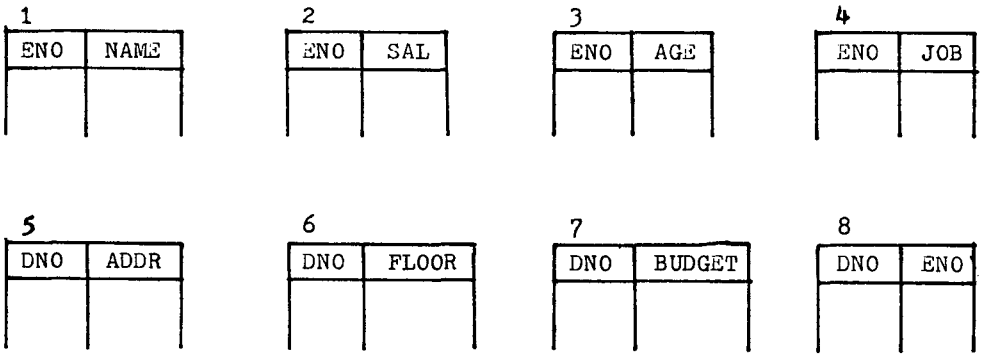

Figure 10. Storing Data as PCPs