THE UNIVERSITY OF MANITOBA

AN ARCHITECTURE FOR A DATA MANAGEMENT SYSTEM

by

JOHN W. REECE

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

WINNIPEG, MANITOBA

OCTOBER 1977

AN ARCHITECTURE FOR A DATA MANAGEMENT SYSTEM

BY

JOHN W. REECE

A dissertation submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ACKNOWLEDGEMENTS

The author gratefully acknowledges the aid and encouragement received from his advisor, Professor R.J. Collens. He also wishes to thank S.A. Bukhari and H. Ferch for their suggestions to improve the clarity of the text. Their contribution, together with that of Mrs. L. Burkowski, was particularly important because of the short time-frame. For the preparation of the manuscript, he thanks Richard McDonald.

The research was financially supported by the Canadian Wheat Board. The author thanks M. Head and F. Jefferson for making the funding possible.

Finally, the author cannot thank his wife enough for her consistent support and baby-sitting throughout preparation of his thesis.

TABLE OF CONTENTS

1 INTRODUCTION

This thesis describes the architecture of a data management system DATA RESOURCE MANAGER (or DRM). The term data management system(DMS) is chosen to designate a complete system of storage, manipulation, definition, and control facilities rather than the simpler physical data base management system (DBMS) which provides the accesing methods of the DMS. DRM was designed to make a high-level data language, useable by non-programmers, available with current data base organizations. In addition, it illustrates that the methodologies chosen to implement these facilities lead to a flexibility which allows migration from current to more advanced technologies as the opportunity arises.

Chapter 2 introduces the data base concept and identifies some of the problems that have limited the success of data base systems. Historical development of the concept is explored from both organizational and technological viewpoints. Three significant emerging trends discussed are:

- management of data as a resource

- defining data in terms of its information content
- a set-theoretic (relational) model of data structure and manipulation

The fundamental purpose of DRM - the goal of developing a technology to complement and exploit these trends - is defined.

Chapter 3 defines objectives for data management systems within a context of management objectives for data. The characteristics of DRM are described and related to the objectives. Where possible, significant alternatives are compared to clarify the selection rationale.

Chapter 4 describes the DRM architecture in detail. Success of a data management system depends upon a powerful end-user language. The Data Resource Access Facility (DRAFT) is introduced. Examples of all types of statements in the stand-alone, interactive part of DRAFT are given. Overall module structure and control flow gives an overview of the system internals. The specific implementation described is a stand-alone

query language over an IMS data base, although the system is equally adaptable to statements imbedded in a batch host language or a differemt data base system. Function and operation of the major modules of DRM are specified. Numerous examples are used to clarify the request translation and execution process.

Chapter 5 summarizes the insights gained from the development of the DRM system architecture. Areas for further study and development of the DRM concept are identified.

2 DATA MANAGEMENT SYSTEMS

There has been a steadily widening interest in data base systems over the last decade. Many corporations have made large investments in data base technology but, alas, few have been rewarded with even a fraction of the benefits attributed to the concept. This chapter examines some of the reasons for this failure and the potential rewards to those organizations remaining optimistic of final success.

2.1 BACKGROUND

The main problem in discussing the data base concept is the lack of common agreement as to what a data base is. (There isn't even agreement to the spelling of the word - see Bibliography.) As a result there has been a divergence of expectations from the reality of technology. Advances in both management's understanding of data and the technology of data base systems will eventually lead to the expected benefits.

To set the scene for the following discussion several terms are now defined. For this thesis a DATA BASE is defined thus:

> A DATA BASE is a collection of interrelated data, stored with controlled redundancy, independent of application programs, to serve multiple requestors. The data base is managed by a single software system (known as a data management system).

Three levels of structures have been defined for

data bases. They are commonly known as <2>:

Internal - the physical structure as stored on disk or mass storage

Conceptual - the overall logical structure as defined by the DBA. (The data base as perceived by the user community.)

External - the subset of the data base defined by a particular user or program.

Three roles are often referred to:

Requestor - Any terminal user or application program which accesses the data base.

Data Administrator - Has overall responsibility for management of integrated data. The DA defines policy and formulates the long-range data base plan.

Data Base Administrator - Is responsible for operational management and control of the data bases. The DBA defines data bases, protects data integrity, and optimizes data base performance.

A widely held misconception has data base systems

maintaining a centralized pool of all enterprise data forming the basis of a total Management Information System. It is likely beyond human capability even to fully understand the scope of the undertaking <15>. There is doubt that it would be desirable to construct a total MIS, were it possible. Decision making is too abstract to identify the inputs to a given decision with certainty. At best the data base can be a source of some of the information to support decision making.

Simply giving the manager more information is not going to improve his decisions. The need is to provide an effective information delivery system. Conventional application files contain much of the data the manager needs. Often this data is unavailable to management due to a lack of relatability of one file to another. The time and cost of developing programs to relate the data may be too great to bear. On the other hand, the information the manager receives is buried in too much data.

There is an emerging trend to treat data as a valuable resource which can be managed by well known principles to maximize its potential value to the

enterprise. Proponents of this concept, known as Data (or Information <40>) Resource Management <37> suggest that achievable expectations for data are that it be:

- manageable like any other resource
- organized to facilitate ad-hoc requests
- capable of systematic growth and adaptation
- integrated across organizational units of the enterprise

With these expectations in mind, chapter 3 develops DRM objectives and requirements in top-down fashion.

In parallel with the evolution of the understanding of data, data base technology has developed from its predecessors: generalized input/output routines and file access methods. Mostly DBMSs have been used as little more than sophisticated access methods and data bases as no more than complex files. Each system is designed in the traditional manner. Data bases are structured in the limited context of the application. Unrelated data bases, with application orientation, have no advantage over conventional files.

To integrate data across several applications, a new approach to analyzing information requirements is essential. The entity concept structures data in terms of the information it represents. This idea is explored in the next sub-section.

The first commercial data base management systems appeared around 1968. Systems in wide use today date from between then and 1972. They are characterized by the data structure that the user sees. They implement one of two models: network<13,8> or hierarchic<28,34>.

In the early seventies, a third structure called the relational model was developed. It possessed a greater conceptual simplicity than the others and had a firm foundation in mathematical set theory. There has been much academic interest in relational data bases since Codd first solidified the concepts in 1971 <12>. A significant body of knowledge exists about the properties of relations and operations upon them. The few implementations <27,30,35> are experimental and use sophisticated physical data structures and access methods. This single technical limitation is a major

factor slowing the acceptance of relational systems.

To date no commercial relational data base system is available. Recently emphasis has shifted from the rigorous definition of properties to practical aspects of implementing a relational data management system. This trend has lead to the DRM concept. Advantages of a relational approach are identified in subsection 2.1.2.

2.1.1 THE ENTITY CONCEPT

Traditionally data structure has been drawn from the design of the application system itself. To develop an integrated, shareable data base, a new source of data organization must be found. The data base is not merely a repository of values, but is a model of some limited universe. Thus our understanding of that universe might be used as the basis for determining data requirements.

The entity concept is a way to look at the nature of information itself. There are three realms when

looking at information and, as Engles <19> noted, we tend to jump from one realm to another without warning. A clear understanding of these realms and their relationship to one another is needed in order to systematize their mapping in the data base.

Firstly there is the real world of objects, people, concepts, etc. which are of interest. These "things" are known as Entities. Employees, products, and bank accounts are entities. Properties are the characteristics of an entity - age, colour, or balance.

The information realm consists of ideas about the real world which exist in the minds of men and women. Entities are represented by property classes called attributes. That Jane Bloggins is a programmer would be represented by a value "programmer" in her POSITION attribute. Our conception of an entity is represented by a collection of attributes, which correspond to the properties of the entity, grouped together to form an Entity Record. A property by which an entity is commonly known (ie "Jane Bloggins") is called an ID attribute. Relationships between entities are also of

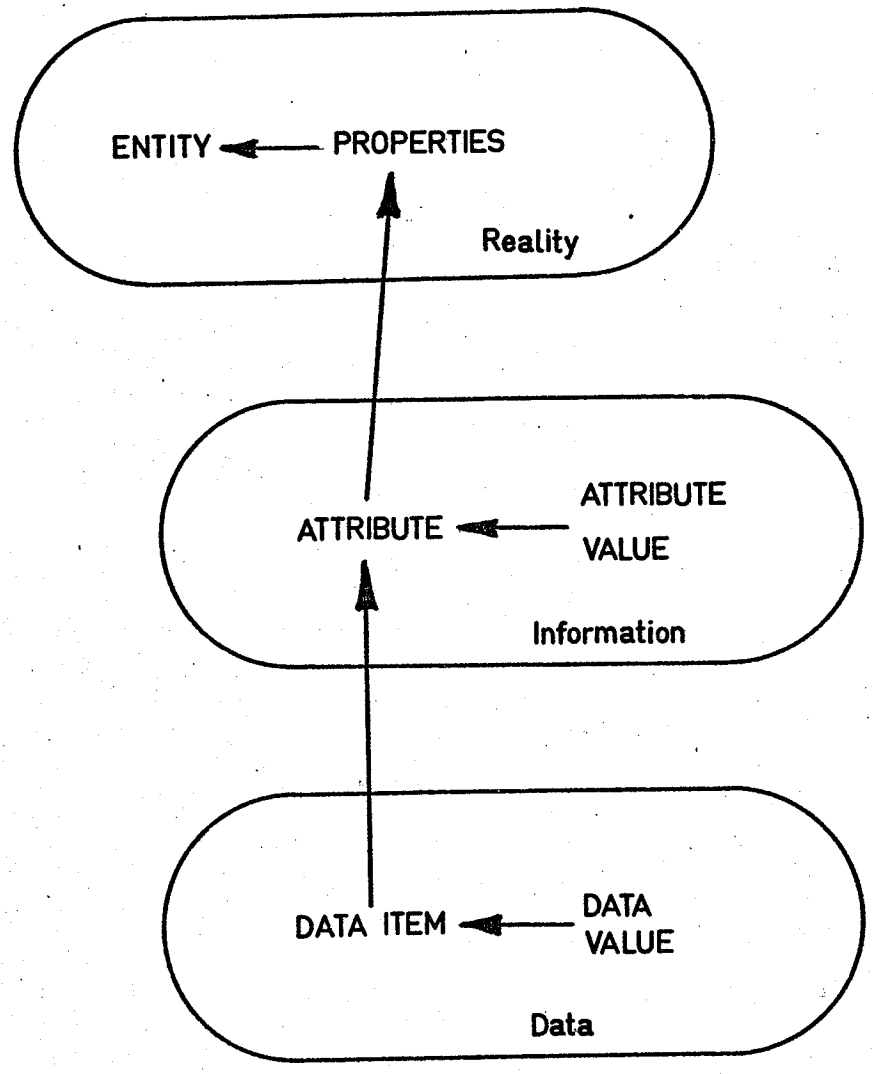


FIG. 2.1 THREE REALMS OF INFORMATION
(after Engles)

interest. The records and relationships relevant to a requestor form an external (or logical) view of the data base.

In the third or data realm are the data items that represent attributes and data values which represent attribute values. There may be many equivalent codings for a given attribute value (ie "2", "TWO", "II"). Values of the ID attribute are represented by key items which are used to access instances of entity records.

Mapping from reality to information is a matter of determining the things of concern to the enterprise and defining entities with attributes corresponding to the properties of interest. The set of these entities which are computerized forms the conceptual data base. This process is largely a matter of judgement on the part of the data base planners. Some care must be exercised in analyzing what is really of interest to an enterprise and, therefore, what the entities are. For instance, is there one PERSON entity in a company's data base or separate EMPLOYEE, SPOUSE, and CHILD entities? (What are the implications of this decision if a father and

son both work for the company?). A set of rules to aid in making these decisions, known as Normalizations, are discussed in the next sub-section.

The conceptual data base is defined to the DMS by statements in the Data Definition Language (see section 4.1). The data realm consists of the corresponding data items and records on physical storage. Mapping between the two realms is defined by the String Model (section 3.3.4). Conceptual records may be materialized or used to form a basis for definition of external views of the data base. Storage and maintenance of physical data and materialization of external records from data is the responsibility of the Data Management System.

2.1.2 THE RELATIONAL MODEL

In its simplest form the relational model is a tabular representation of data. A table corresponds to an entity. Each row is an instance of the entity and each column is an attribute of the entity record. For example the EMPLOYEE relation shown as a table in

EMPLOYEE

NAME	POSITION	DEPT	SAL
ROSS	PROGRAMMER	4	125
DOE	LIBRARIAN	4	100
LEE	PROGRAMMER	4	150
SMITH	CLERK	9	80

DEPARTMENT

DEPT	MANAGER
9	WILLIAMS
2	JONES
4	HANSEN

E1 List all employees who are programmers.

NAME	POSITION	DEPT	SAL
ROSS	PROGRAMMER	4	125
LEE	PROGRAMMER	4	150

E2 List all employee's names and salaries.

NAME	SAL
ROSS	125
DOE	100
LEE	150
SMITH	80

E3 List all employees and their department.

NAME	POSITION	SAL	DEPT	MANAGER
ROSS	PROGRAMMER	125	4	HANSEN
DOE	LIBRARIAN	100	4	HANSEN
LEE	PROGRAMMER	150	4	HANSEN
SMITH	CLERK	80	9	WILLIAMS

FIG. 2.2 EXAMPLES OF OPERATIONS ON RELATIONS

Figure 2.2, where each row is a person and the columns specify the employee's name position, etc.

The approach has a set of simple but very powerful operators to query and manipulate tables and entries. Results of all operations are, themselves, tables which may be displayed or further operated upon. In practise the most common operators are: Selection, Projection, and Joining.

Selection extracts a subset of the rows of a table which satisfy a given condition. Example E1 shows the result of selecting all programmers from the EMPLOYEE relation.

Projection extracts a subset of the columns from a table as specified by a list of column names. Example E2 shows the projection of the NAME and SAL columns from the EMPLOYEE relation.

Join concatenates rows of two tables which share a common value in a column of both tables. In example E3 rows of EMPLOYEE and DEPARTMENT are joined when their

DEPT columns are equal. Note that one of the duplicate columns is deleted.

The relational approach includes guidelines for data base design called Normalizations <12>. In the previous section it was stated that it is desirable to structure the data according to its "meaning". Records should correspond to a single entity, with one concept described by each relation.

First Normal Form restricts the attributes of a relation to simple values. Multi-value properties are separated into a new relation. The many projects that an employee works on are not in the EMPLOYEE relation but in a PROJECTS relation, which may have as many instances as "projects" an "employee" is assigned to. Second Normal Form requires that all attributes must correspond to a property of the entity represented by that relation. "Name" could not be stored in the PROJECTS relation because it is a property of the "employee" only and not the projects assigned to him. Third Normal Form requires that only attributes that are directly related to the entity be in the relation.

That is, "manager" is a property of an EMPLOYEE, but only indirectly so - by association of an "employee" with a "department". The "manager" should only be an attribute of a DEPARTMENT relation.

The inconsistencies of access and storage which normalization eliminates can be illustrated by an EMPLOYEE relation with a "manager" attribute. If all employees of a department are reassigned the record of who is department manager is lost. Moreover, how would the DMS go about answering the question: Who is manager of the sales department. Finally consider the redundant effort to change the manager for a given department.

To perform the various relational operations, it is required that relations be at least in First Normal Form. This is no barrier to using a relational view of a conventional data base, since the normalizations are equally applicable to design of hierarchic or network data structures <38>. (DBTG repeating groups could not be used in the structures because they contradict the first normal form.) In any case, the rules do not guarantee a good design since they are only concerned

with a structure consistent with the defined entities not the "correctness" of the designer's understanding of enterprise entities.

It is shown in section 3.3.3 that the conceptual simplicity of relational systems (in particular, the single storage and access constructs) will facilitate a greater degree of user independence from physical considerations than is achievable with the other data models.

2.2 DATA RESOURCE MANAGER APPROACH

DRM is an architecture to illustrate how advantages of a relational external structure can be realized within the context of current technology. This thesis proposes that these advantages are mainly associated with independence of the user from physical structure, and therefore can be achieved by providing, in the short term:

- a relational external view over a conventional data base structure
- a relationally complete <11> access language useable by the casual user

The system will be developed with an inherent flexibility to allow evolutionary growth to a complete package providing:

- protection of current investment in data and programs
- the ability to accomodate changes in requirements or users
- the freedom to modify the internal environment to optimize performance

System specifications were developed within the

context of a set of widely accepted objectives for data management systems. These objectives in turn relate to management expectations for data.

Methodologies and algorithms selected for the DRM functional modules all have a sound theoretical basis. They have been united into a facility which provides a complete range of operations with reasonable system performance, while maintaining the independence of users from data which will allow DRM to adapt to remain viable for a significant time.

3 SYSTEM REQUIREMENTS

Management perspective is maintained in the definition of the architecture for DRM by developing the functional requirements for the system in a "top-down" fashion, the top in this case being the expectations general managers have for information. These are stated in terms of requirements for resource management, and specifically, implications for the data resource.

Characteristics of generalized data base management systems (DBMS) are reviewed in the context of management expectations. Seven requirements, defined by a joint committee of the GUIDE and SHARE computer users groups, are adopted because of their consonance with higher level objectives and almost universal acceptance within the industry.

The functional specification of DRM is detailed in the final section of this chapter. Each feature of the system is defined and justified in terms of the GUIDE/SHARE requirements. Alternative approaches are

used to illustrate the selection rationale.

3.1 MANAGEMENT EXPECTATIONS FOR DATA

Treating data as a resource puts information processing in terms that general management understands. With this understanding comes a greater appreciation of the value and limitations of data. Resource management orientation will result in expectations for data which parallel those for other resources such as cash, material, or personnel. In general they are:

Useability	- ability to apply resource to achieving enterprise purpose.
Controlability	- ability to insure validity of and control access to resource.
Efficiency	- optimization of costs and exploitation of resource
Adaptability	- protection of investment and ability to sieze opportunities.

The following sub-sections consider these with specific reference to the data resource.

3.1.1 USEABILITY

The usefulness of data is a function of the ability to derive relevant information from the data base. Relevance is determined by several qualities whose value varies with each circumstance.

To begin with, the data items which satisfy a request must exist in the data base in an accessible form. The organization of the data determines the derivable information content. In many conventional file systems, all elements to satisfy a request exist but are so fragmented that it is not possible to extract the answer economically. The DBMS must provide facilities to organize the data such that information available from a given body of elements is maximized.

Information must be delivered to the requestor in a format and of such a volume so that he can understand

it. Most managers are buried in volumes of data too great to be absorbed. Information must be timely. Month-old reports cannot support day-to-day decisions. Information must be credible; the manager must trust its validity.

Relevance is also dependent upon the level of the requestor in the organization. Robert Anthony <1> has classified three levels of managerial activity:

- strategic planning
- managerial control
- operational control

Each is characterized by distinct decision rules and, therefore, different information requirements. The first two are general management activities which demand a corporate-wide perspective. Their requests for information typically cross functional boundaries. The data base must integrate the data of the various functional units into a global structure matching their perspective.

3.1.2 CONTROLABILITY

Within this context control will be defined as the evaluation of decisions. Activities involved are:

- determining performance indicators
- monitoring actual performance
- comparing planned to actual performance
- taking any necessary corrective action

For data management typical key results areas include system response, data validity, and security. The DMS must provide mechanisms to parameterize system reponsiveness, insure the integrity of data, and control access to the data base. The system must also have built-in monitoring capabilities for measuring actual results.

The data base administrator must be in a position to modify the physical structure as a corrective action. User programs should not be impacted by the modifications.

3.1.3 ADAPTABILITY

Change is the only constant in the data environment. Adaptation is a response to change which minimizes potential reductions in efficiency. The data management system should be able to adapt to change in two ways:

i) Accomodate new requirements and users without impacting old users.

ii) Reorganize or take advantage of new technology to improve efficiency.

3.1.4 EFFICIENCY

Once the purpose of an organization has been determined the fundamental business problem is to optimize the conversion of resources to achievement of that purpose. When corporate information requirements are defined, the data administrator is left with implementing the data structure which satisfies these requirements at minimum cost.

The data management system must allow for the ongoing monitoring and tuning of the data bases. Structural changes to improve performance must be possible without adversely impacting any current users.

3.2 DATA MANAGEMENT OBJECTIVES

Rather than focusing upon a particular application, the DRM system is intended as the software manager of all data bases of the enterprise. Therefore, it must be capable of serving the often conflicting information needs of all areas of the enterprise. To warrant implementation they must be capable of performing this function across a broad spectrum of organizations - government, business, universities.

It was a group of such broad representation - over forty organizations - that were brought together in a joint project by the computer users groups GUIDE and SHARE to determine the long-range requirements for data management systems. Their report <24>, tabled in 1970, represented a comprehensive set of demands for features in future data base management systems.

Each of the GUIDE/SHARE requirements has been the subject of continued interest and extensive research. Since the objective of this paper is to bring together the results of the most promising of these independent

research efforts into a unified system architecture, it is reasonable to adopt the following seven requirements for a data management system:

- Program/Data Independence
- Data Relatability
- Non-Redundancy
- Integrity
- Security
- Performance
- Compatibility

3.2.1 PROGRAM/DATA INDEPENDENCE

Programs must be able to access data independent of the internal storage structure of the data even as that structure evolves to accomodate new changing usage. Three types of change are inevitable in the overall data environment:

- growth of the data base
- new users
- technological advances

Data management systems must be free to adapt to these changes while preserving the viability of current programs.

When a program is designed its functions are implemented according to an understanding of how the data it requires will be accessed and presented. This user data structure is usually referred to as an external "view" of the data base. Protection of the investment in that program is a matter of maintaining the ability to deliver data to the program in the form expected by the program.

A typical program requires access to only a subset of the elements of the data base. External views are windows through which the program views this subset - the data base as the application sees it. To achieve independence program operations must be definable in terms of the logical view. Since this logical data base is simpler than the underlying physical structure, coding of applications is simplified.

Independence gives the DBA greater freedom to manage

the data base effectively. Since programs are no longer bound to a particular structure, reorganization can be done as required to optimize performance or to accomodate new users and data elements. Greater control of the data base can be exercised by limiting user views to a "need-to-know" set of data elements.

Several classes of data independence are identifiable. It is useful to consider each as a further aid to understanding the independence concept and to establish some key indicators for the evaluation of DRM. Leon Smith <42> presents five classes, of which the first two (storage device and physical record independence) are features of all modern access methods. Lumping them with the third yields three major classes:

- physical structure independence
- representation independence
- data structure independence

The actual performance of a data base system is a function of the physical organization of the data in secondary memory (disk, mass storage). Physical

structure independence exists if programs are isolated from changes to that organization. Data management systems should provide several access methods with different performance characteristics but with equal capability from a user viewpoint. The DBA may select from these alternatives to optimize performance.

A representation is an encoding of the symbols by which we identify some object fact or idea. "John Jones", "DP Manager", & "611 473 061" might all be valid representations of the same person. A quart is also 1.14 litres and might be represented that way in a system developed after metrification. The system must serve this new system along with older systems which work in quarts. Representation independence is the ability of the data management system to convert the stored value of an item to an equivalent coding as defined by each user of that item.

Data structure is the combination of elements into segments, segments into records and relationships between segments and records. Many equivalent physical representations exist for a given structure. Various

models of data structures are considered more fully in sections 3.3.1 and 3.3.4 . Evolution of the data base and growth of the user community result in changes to the global data structure (typically, addition of data elements or relationships or splitting of records). Data structure independence is the capability to preserve the external data structure viewed by current applications as the internal data structure changes.

3.2.2 RELIABILITY

Much of the information content of a data base consists of the relationships between data elements. Where these relationships are of interest to a requestor the data management system must provide the capability to specify the associations completely and unambiguously and to extract occurences of a relationship from the data base.

The problem facing the DBA is how to satisfy the needs of a diverse community of requestors who, in general, have many different concepts of what is

relevant in the data base. It was shown that an object of interest is viewed by a requestor as a collection of properties which can be represented by a group of attributes forming an entity record. An example would be an EMPLOYEE entity consisting of SIN,NAME,POSITION, and MANAGER attributes. Relationships between entities (for instance EMPLOYEE and DEPARTMENT) are also of interest. The records and relationships relevant to particular requestor form an external view of the data base.

Various external views are combined by the DBA into a global data structure. Since the views, in general, overlap one another, the resultant internal data mapping is complex. The data management system must preserve all of the external relationships in the internal structure. A declarative mechanism is provided so that the record and relationship types can be defined to the system.

Specific instances of the relationships must also be preserved. It is not sufficient to know simply that an EMPLOYEE is related to a DEPARTMENT. The DMS must be

ENTITY TYPES

EMPLOYEE (SIN,NAME,POSITION,MANAGER)

DEPARTMENT (NAME,LOCATION)

ENTITY INSTANCES

(611 492 075, REECE, SE, LEROUX)

(692 999 431, CZUK, PROG., REECE)

(SYSTEMS, EMPRESS ST)

(MARKETING, PIONEER AVE)

(PLANT, CORYDON AVE)

FIG. 3.1 THE TYPE / INSTANCE DISTINCTION

able to record that EMPLOYEE("JONES") works in the DEPARTMENT("SALES"). Figure 3.1 illustrates the difference between types and instances.

3.2.3 NON-REDUNDANCY

In an integrated data base there exists potential for duplication of data items. This leads to excessive storage costs and, as shown in discussing normalization, inconsistencies and difficulty in updating the data base. When several versions of an item are in the data base discrepencies may occur between versions. Credibility of the data disappears if it is not possible to determine which is the correct value.

The data management system must be able to meet the requirements of relatability with "minimum" redundancy. The previous statement differs somewhat from the GUIDE/SHARE call to "eliminate" redundancy. Some redundancy may be necessary for the preservation of relationships or desirable from a performance

optimization standpoint. In any case, the data management system should be responsible for maintaining redundant values automatically without requestor intervention.

Casey & Delobel <4> put forward the premise that a data base was analogous to a switching function and was, therefore, subject to optimization according to algorithms for minimization of switching networks. Redundancy is reduced by defining a data structure where redundant data is replaced by relationships. By applying their methods the number of relationships required to be stored is reduced to a "minimal cover". A minimal cover is the fewest relationships from which can be derived all original relationships (know as the transitive closure). The DMS must be capable of generating any relationship in the closure from a stored minimum cover.

Similar methods have been proposed by Wang & Wedekine <44> for the minimization of redundancy in external view design.

3.2.4 INTEGRITY

A data base is not merely a collection of data items, but is a model of some limited universe. It is said to have integrity if it represents a reasonable configuration of that universe <26>. Protection of integrity consists of taking steps to ensure that the values and relationships of the data base are not damaged or see that values conform to user or DBA defined rules of legitamacy. Potential threats to integrity include computer system failure, interaction of concurrent accessors, and erroneous input data. Facilities of the DMS will be designed to prevent damage, but since some damage will enevitably occur there must also be provision for recovery.

Prevention falls mainly into two categories:

- isolation of concurrent users
- enforcement of legitimacy rules

In order to isolate users, a locking mechanism must be employed. Locking is the process of granting exclusive use of some subset of the data to a

subprocess. Two properties characterize any scheme for locking. First, the mechanism is either high-level <20>,(determining which requests may run concurrently by comparison of their target items and selection criteria), or low-level <16> (enqueing upon actual items or groups of items in the data base). Second, the mechanism is either explicit (the requestor specifically identifies the subset of the data base to lock) or implicit (the DMS shoulders the burden of locking transparent to the requestor).

Whenever locking is allowed there is potential for deadlock, a state where sub-process A locks subset S, and requests a lock on S2 while sub-process B has locked S2 and requests S1. A circular wait has been created where neither sub-process can complete. Three mechanisms are commonly employed to resolve deadlock:

- prevent conflicting requests from executing concurrently.
- avoid executing potentially conflicting requests.
- detecting deadlock and backing out one of the conflicting requests.

Legitamacy rules are enforced by execution of a user defined function (often called an edit routine) whenever a related state change occurs in the data base. These functions are predicates, defined to the system independently of any application programs or request blocks. Whenever a set of conditions, which form part of the function definition, exist the function is invoked. If the predicate evaluates TRUE then execution continues, otherwise execution terminates and an exception condition is returned to the requestor.

When damage occurs some means of recovering the data base to a legitimate state must be invoked. The actual process of changing a value in the data base requires execution of many machine instructions. Should a system failure occur during this process the data base may be inconsistent. The DMS must be able to back-out the effects of the process so that the data base is again consistent. A journal or log maintained by the DMS will be used for back-out.

Sometimes the damage is so extensive that the data

base must be completely recreated. In order to do this occasional copies of the data base are made and used as the source for recovery. The DMS examines the journal and applies valid changes to the restored data base until the last consistent state, prior to the disaster, is reached.

3.2.5 SECURITY

Data security refers to protection of data against accidental or intentional disclosure to unauthorized persons, or unauthorized modifications or destruction <32>. There are numerous aspects of an adequate security program including physical protection, identification devices, and personnel reference checking but this discussion will restrict itself to those factors implementable within the DMS.

The requestor must be positively identified to the system before accessing the data base. A user-id known to DMS is supplied before requests are acted upon. The requestor must be authorized to perform the operation

requested. Type and target of each request is compared to the user's profile, which is associated with his user-id. The requested operation will be performed only if the user has authorization.

An audit trail, recording all access to the data base with the requestor's ID, is maintained on the journal. Responsibility for every action will be known by the responsible authorities.

3.2.6 PERFORMANCE

Regardless of whether the system is to operate in on-line or batch mode, it is important that the system respond in an acceptable time at reasonable cost. What constitutes "acceptable" and "reasonable" is determined by circumstances and user attitude. Except for specific, dedicated environments, evolution of the data base and user community will cause performance factors to change with time. In the section on independence it was stated that the DBA should be free to take whatever actions are necessary to establish and maintain

satisfactory performance.

In addition to some degreee of independence, the DMS must provide options of structure, accessing mechanisms, and request parameterization which differ in their performance characteristics. The DBA selects the combinations of options which optimizes performance for that particular environment. Monitoring facilities enable him to evaluate system operation continuously. As tuning becomes necessary, changes shall be implementable without incurring a prohibitive cost.

3.2.7 COMPATIBILITY

An enterprise's investment in programs and data must be protected. On the other hand, advances in technology, both at the requestor interface ind in physical storage devices, offer benefits that should be realized. Compatability is being able to develop applications using new request languages to access current data bases and continue to execute old programs after the data bases have been migrated to new storage

structures.

Since migration takes some time to occur (in particular, old programs never seem to die) the DMS must support many combinations of new and old technologies concurrently.

3.3 DATA RESOURCE MANAGER CHARACTERISTICS

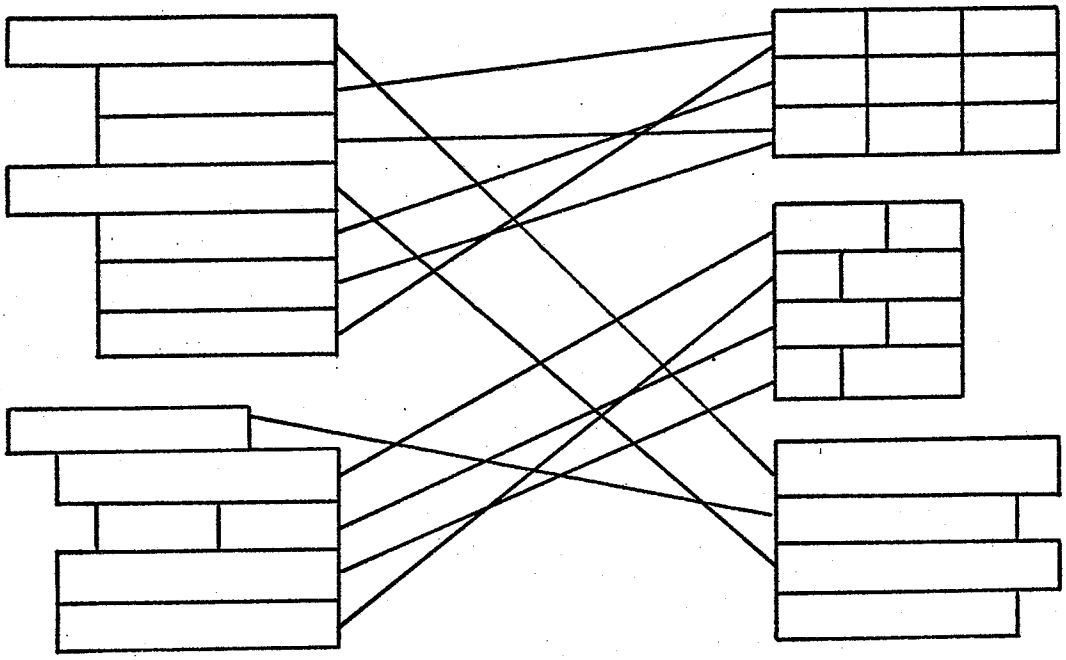
The highest level functional properties of DRM are described in this section. Together they form the external architecture of the system - the "must" functions - that achieve the requirements set out in previous sections.

Each characteristic is described in terms of what lead to development of the concept, detailed studies of the concept, and its relationship to the requirement areas.

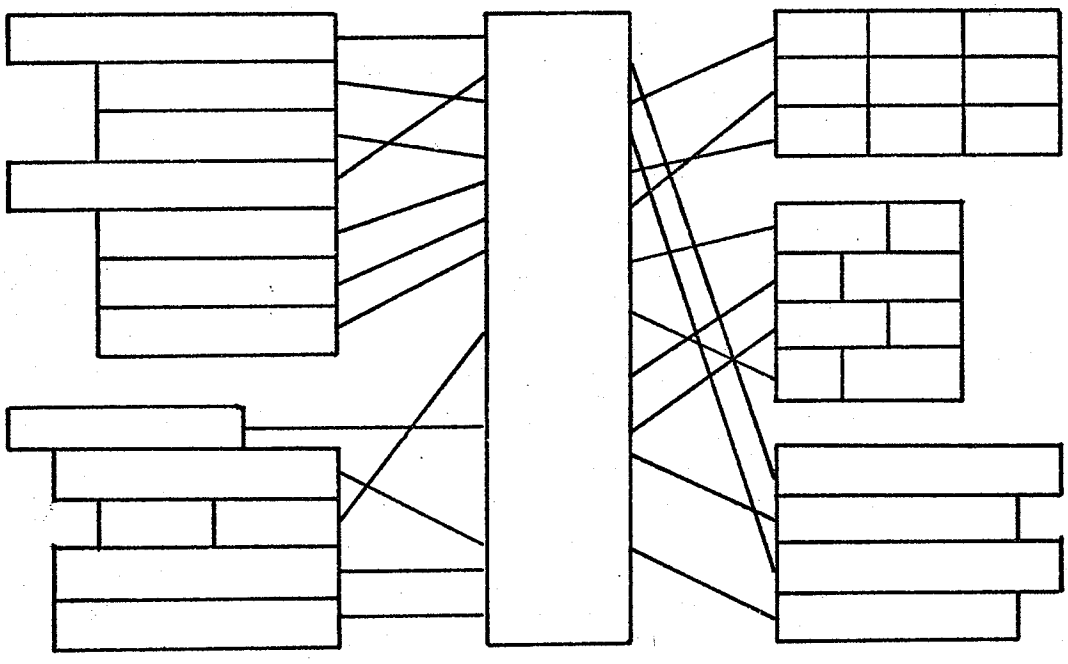
Where possible the several alternatives are compared as an illustration of the selection rationale.

3.3.1 THREE LEVEL DATA STRUCTURE

Any true data management system offers at least a two-level data structure: physical data base and application program logical data. (see fig 3.2-a). Most commercial systems today (IMS<28>, IDMS<13>) have



A. TWO-LEVEL STRUCTURE



B. THREE-LEVEL STRUCTURE

FIG. 3.2 MULTI-LEVEL DATA STRUCTURES

exactly two levels. Since the two levels of structure must interface directly at least one structure must be defined in terms of the particulars of the other structure. This violates the requirement for independence of the external and internal structures from one another. A solution is to adopt an intermediate level of structuring, often called the global data description, which will serve as a stable model to which the other two models are related. (fig 3.2-B)

The three-level structure was first proposed by the GUIDE/SHARE joint committee, where the intermediate level was known as the Conceptual Model. Thorough discussions appear in a paper by Mairet<31> and one of Martin's books <33>. As a first step to standardization of data base systems the Standards Planning and Requirements Committee (SPARC) of the American National Standards Institute (ANSI) has proposed a model architecture encompassing the three-level structure <2>. In spite of this direction, and perhaps as a result of the self-evident nature of the concept, it has received little attention from the

theorists. It is, one might say, left as an exercise for the implementor to perform.

Three functions are identifiable for support of a three-level structure. They are:

- Conceptual Data Description Language
- Mapping from External to Conceptual Model
- Mapping from Internal to Conceptual Model

Recognizing that the requestors understanding of the data base will be in terms of the Conceptual Model. The External and Conceptual Data Descriptions of DRM are combined into a single unified facility, together with the Data Manipulation Language. Mappings between the two levels are accomplished by means of statements in the DML. This approach is explored more fully in section 3.3.3.

In order to allow the maximum flexibility in defining the internal storage structure, the mapping between the Conceptual and Internal Models must be capable of describing arbitrarily complex structures. A string modelling method used by DRM is described in

section 3.3.4.

3.3.2 SELECTABLE INTERNAL STRUCTURE

What constitutes the optimum internal structure for a data base will be determined by a complex of environmental considerations which will vary significantly both between enterprises and with time within a given enterprise. The DBA must have the freedom to select the characteristics of the internal data base to achieve the optimum.

In particular DRM must allow access to the current data bases with the full power of its user language and the migration of those data bases to new technology structures while old programs remain viable.

Three commonly accepted models of data structuring have been implemented in practical data management systems. With some examples, they are:

Network - IDMS, DBTG<8>

Hierarchic - IMS, S2000<34>

Relational - System R <3>

Data structures must model both data and relationships. Each model type will be described in terms of the facilities to describe records and relationships.

NETWORK MODEL

In a network data structure any record type may be connected to any other record type. Relationships are defined by named paths connecting the related record types. The data mapping is described as complex, that is one-to-one, one-to-many, and many-to-many relationships are possible.

For practical purposes, systems do not implement many-to-many relationships directly, but introduce an intermediate record with one-to-many relationships to the other records. Systems employing this restriction recognize two types of records: owners and owned. A record type may be an owned record in some relationships and, at the same time, an owner record in

others.

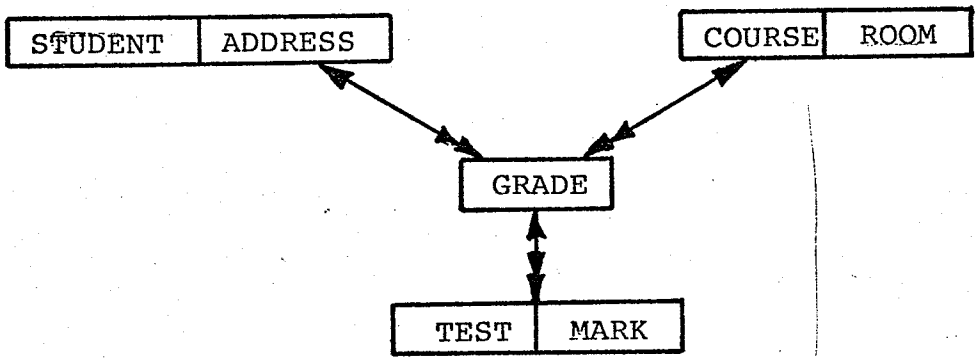
An example of a network data structure is shown in figure 3.3 a.

HIERARCHIC MODEL

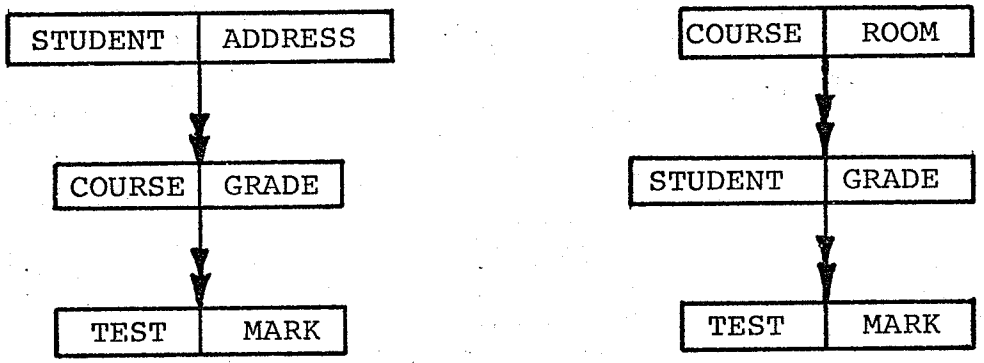
In a hierarchic data structure a record type may be the owner of many record types but may only be owned by a single record type. Owners are known as parents; owned records as children. Relationships are represented by a path connecting the related record types.

If a many-to-many relationship is to be modelled, it is described with two hierarchies each representing one of the one-to-many relationships which are, together, equivalent to the original relationship.

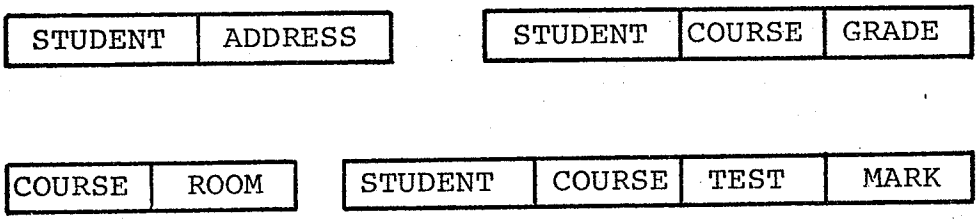
An example of a hierarchic data structure is shown in figure 3.3 b.



A. NETWORK



B. HIERARCHIC



C. RELATIONAL

FIG. 3.3 DATA MODELS

RELATIONAL MODEL

In a relational data structure records are never connected together. Instead relationships are described by additional record types with compound keys (a catenation of the keys of the related record types). Since the catenation may be any combination of the individual key values, many-to-many relationships can be represented. An example of a relational data description is shown in figure 3.3 c.

In order to minimize redundancy, it is desirable to represent relationships by means of links whereever possible. All relationships are represented by links in a network. Hierarchic structures require redundant data when representing many-to-many relationships. Relational structures represent all relationships by redundant keys. From a redundancy viewpoint, the network model is best for internal data structuring.

For reasons of compatability (existing data) or performance (redundancy to reduce access time), it is necessary to support all internal models, selectable at

the option of the DBA. Implementation of this feature will be explored in greater detail in section 3.3.4.

3.3.3 RELATIONAL EXTERNAL MODEL

While performance is of primary concern in the internal realm, conceptual simplicity, consistency and completeness of the model are the most important characteristics of the external model. For now completeness may be defined as the ability to model the three relationship classes. Since this is true of all the models considered in the previous section, they remain candidates for further consideration.

As a next step the complexity of each model is evaluated by determining the numbers and kinds of structural constructs necessary to describe and language constructs necessary to access its data structures. Results of this comparison are summarized in fig 3.4.

The only structural construct in the relational

	NETWORK		HIERARCHIC		RELATIONAL	
STRUCTURAL	RECORDS LINKS	2	SEGMENTS POINTERS	2	RELATIONS	1
LANGUAGE	GET GET ABOVE GET BELOW GET VIA	4	GET GET ABOVE GET BELOW	3	GET	1

FIG. 3.4 DATA MODELS CONSTRUCTS

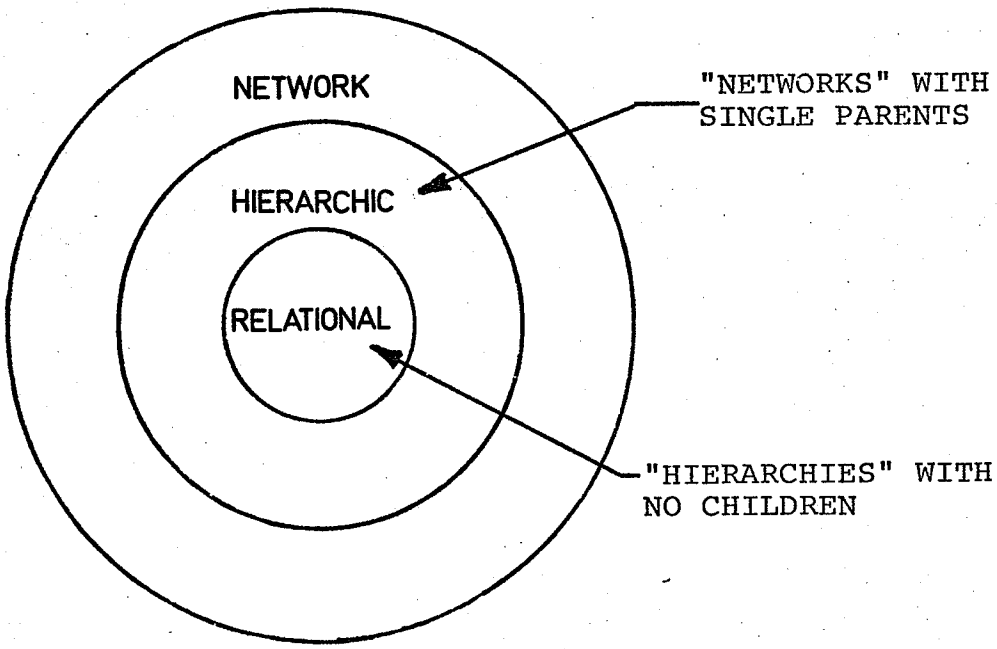


FIG. 3.5 RELATIONSHIP OF MODELS

model is the record (known as a relation). Both hierarchies and networks have two constructs: records and links.

A useful vehicle for comparison of the operations upon the three models is a proposal by Date <14> for unified extensions to high-level languages allowing access to all models. His approach is based upon an intuitive understanding of relations as a subset of hierarchies, which are, in turn, a subset of networks (fig 3.5). The premise is tested by comprehensive examples of the same problems with each model. Results of Date's findings are summarized in figure 3.4. Only one access operator was required in the relational examples and the overall code was correspondingly simpler than its hierarchic or network equivalents.

The relational model is conceptually the simplest of the three models. A relation may be thought of as forming a table, a very familiar method for the display of information. (Tables form the basis of a graphic data language known as Query-By-Example <46>). Columns of the table represent the attributes and rows the

occurences of the relation.

Operations upon relations are not only simple but powerful. Only six (SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE) are required and each can operate upon an entire relation at a time. The power results not only from the scarcity of operators and their scope, but also from the fact that the full functionof the manipulation operators can be employed in declaratives. For instance, since the result of any operation is a relation, statements of the Data Manipulation Language can be used to define external "views" of stored relations. A relational system retains a high degree of independence since operations are not formulated in terms of access paths or specific implementations of links.

The greatest value of the relational model is a consistency throughout its facilities resulting from its origin in mathematical set theory. The possibility of a set theoretic structure was postulated as early as 1962 <9> and proven feasible by Childs <7> in 1968. Codd established a firm foundation for development of

the concept with three papers, published in 1970 - 71. The first <10> defined n-ary relations in a data base context, described operations to manipulate relations, and introduced the concept of normalization. Further normalizations, which simplified the data base design process, were developed in the second <11>; while a rigorous definition of completeness of data base sub-languages was proposed in the last paper <12>.

Since then great interest has been shown in the academic community and the research laboratories of at least one major computer manufacturer. There is now a body of knowledge on most aspects of data management systems based upon the relational model. Examples in various subject areas include works of the following: Languages - Chamberlin <6> and the GUIDE project on high-level languages; Design - Casey <4> and Reece <38>; Performance - Hall <25> and Smith <41>; Integrity - Eswaran <20,21> and Hammer <26>; Security - Chamberlin <5>; and Implementation - Myopoulos <35> and Astrahan <3>.

It is likely that relational systems will become the

dominant force in data management during the next decade. Current implementations are limited to systems with specialized storage structures (System R <3> is typical). The philosophy of DRM is to make the benefits of a relational language available to the general user immediately. Independence and compatibility will allow migration to new storage technologies as they become available.

3.3.4 STRING MODEL

If several internal structures are possible in DRM, a generalized mapping from the conceptual to internal models, capable of describing those structures, must exist. Earley's V-graphs <18> and Senko's String Models <39> provide such mappings.

String Models are chosen for DRM because of two differences giving them more descriptive power than V-graphs. First, the destinction is made between types and instances of strings (Fig 3.6), allowing superior descrimination when selecting the optimum access path.

TYPES OF STRINGS

A1	A-string	EXL=(NUM,NAME)
A2	A-string	EXL=(NUM,PROJECT,HOURS)
E1	E-string	EXL=(A2)
L1	L-string	EXL=(A1,E1)

INSTANCES OF STRINGS

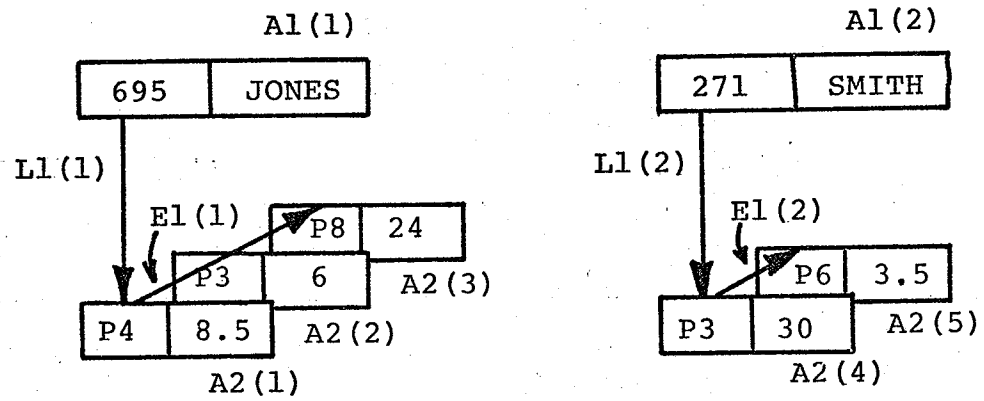


FIG. 3.6 TYPE / INSTANCE DISTINCTION FOR STRINGS

Secondly, string level semantics deal with objects in the conceptual realm (i.e. attributes in particular roles) and are, therefore, independent of the physical structure and access methods.

The three types of strings used in string modelling are:

- A(atomic)-strings connect attributes of a single entity. They are analogous to segments in IMS or records in a DBTG system.
- E(entity)-strings connect subsets of strings, with the same type description, defined by boolean expressions They are analogous to twin chains in IMS.
- L(link)-strings connect instances of strings with different type descriptions They are analogous to child pointers in IMS or links in DBTG systems.

Together these types form a powerful facility to

describe generalized data structures. The string model of the data base used in section 4.3 is graphically superimposed on the IMS physical structure in Figure 3.7. Heavy boxes represent the IMS segments and the heavy arrows the pointers. The equivalent strings are shown as light arrows, except for L-strings which are indicated beside their equivalent Parent-Child pointer. Ae is the A-string modelling the EMPSEG segment and ET is equivalent to the chain linking "twin" occurences of TIMSEG segments.

The string catalogue portion of the Dictionary for the data base is shown in Figure 3.8. TYPE defines the string model function of the NAME. A ROLE_NAME is a synonym for an attribute. ON indicates which strings an entry appears on. The EXL or Exit List indicates the entry types found on this string (i.e. Ae strings lie along Ea strings). SC is the String Criteria and defines which subsets of data are found on an instance of a string. all instances of AS with a given value of DEPT lie on one instance of ES. An instance of Lpa connects an instance each of Ap and Ea with a common value of PROJ.

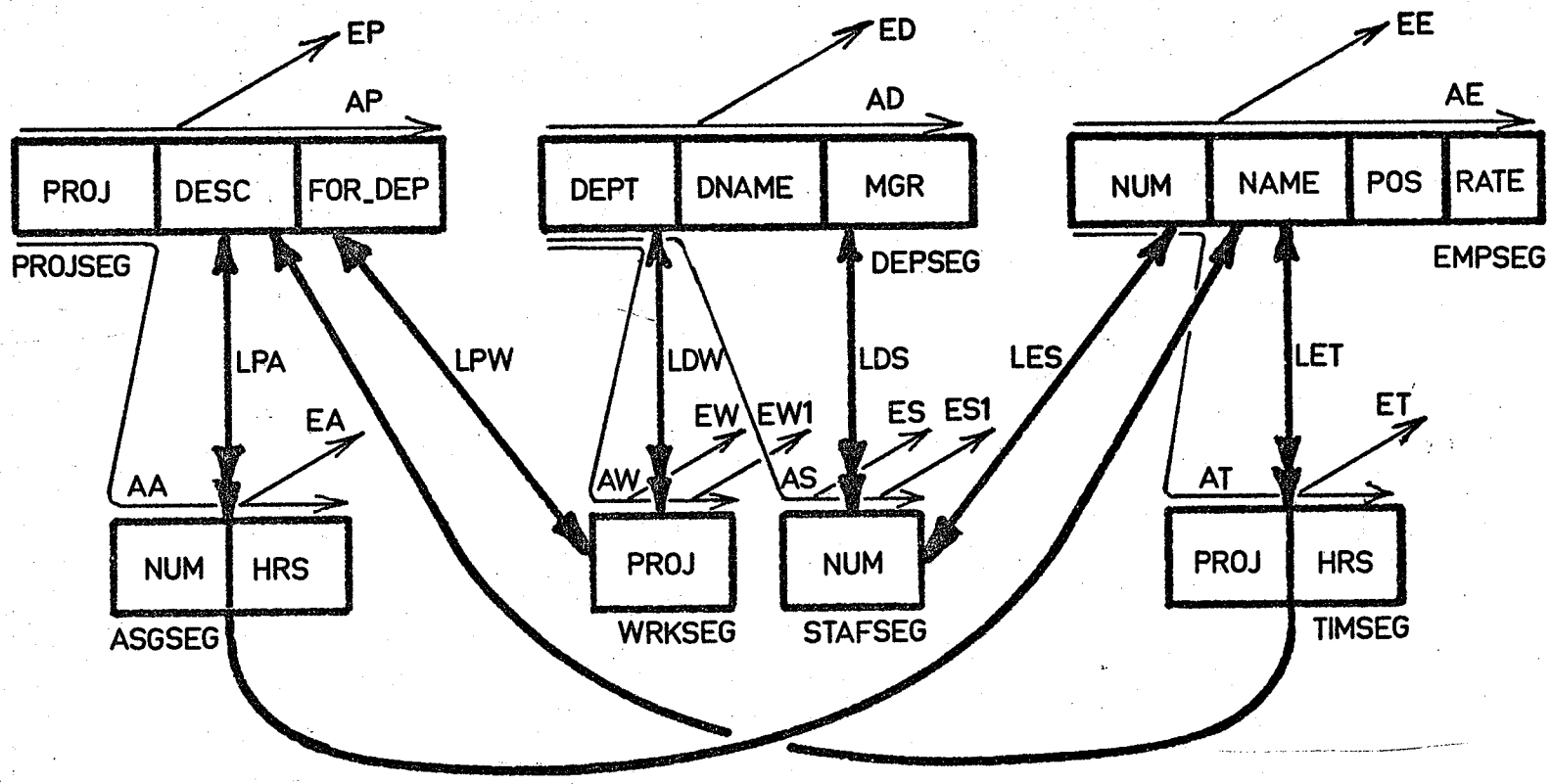


FIG. 3.7 STRING MODEL OF IMS DATA BASE

NAME	TYPE	ON	EXL	SC
HRS	ATTRIBUTE	AA,AT		
PROJ	"	AP,AW,AA,AT		
DESC	"	AP		
FOR_DEP	"	AP		
DEPT	"	AD,AW,AS		
BY_DEP	ROLE NAME	DEPT		
DNAME	ATTRIBUTE	AD		
MGR	"	AD		
NUM	"	AE,AT,AA		
NAME	"	AE		
POS	"	AE		
RATE	"	AE		
AP	A-string	EP,ET	PROJ,DESC,FOR_DEP	
AA	"	EA	PROJ,NUM,HRS	
AD	"	ED,EW1,ES1	DEPT,DNAME,MGR	
AW	"	EW,EW1	DEPT,PROJ	
AS	"	ES,ES1	DEPT,NUM	
AE	"	EE,ES	NUM,NAME,POS,RATE	
AT	"	ET	NUM,PROJ,RATE	

FIG. 3.8-A STRING CATALOGUE (PART 1)

(continued)

NAME	TYPE	ON	EXL	SC
EP	E-string	Ø	AP	OO=PROJ
EA	"	LPA	AA	PART=PROJ
ED	"	Ø	AD	OO=DEPT
EW	"	LDW	AW	PART=DEPT
EW1	"	LPW	AW	PART=PROJ
ES	"	LDS	AS	PART=DEPT
ES1	"	LES	AS	PART=NUM
EE	"	Ø	AE	SEL=NUM
ET	"	LET	AT	PART=NUM
LPA	L-string	EP	AP,EA	MC=PROJ
LDW	"	ED	AD,EW	MC=DEPT
LDS	"	ED	AD,ES	MC=DEPT
LET	"	EE	AE,ET	MC=NUM
LES	"	EE	AE,ES1	MC=NUM
LPW	"	EP	AP,EW1	MC=PROJ

FIG. 3.8-B STRING CATALOGUE (PART 2)

3.3.5 CONTROL SEPARATE FROM ACCESS

Control facilities enable the DBA to define and exercise authority over access to the data base and to implement integrity protection functions. Within the scope of that definition are statements to:

- define relations and views,
- limit access of subsets of data to certain users and vice-versa,
- grant types of access (read, update, etc),
- assert integrity rules,
- trigger secondary functions, and
- revoke privileges.

In order that the DBA maintain control of his jurisdiction, it is necessary to separate control functions from the accessing functions of the system. Controls are defined by declaratives and saved in the Dictionary. When a request is made, DRM invokes all related controls automatically. Failure of any control predicates will cause the request to be terminated and an exceptional condition returned.

3.3.6 DICTIONARY DRIVEN

DRM maintains a catalog that describes all of the objects - relations, attributes, views, controls, and mappings - of the system. This Dictionary is stored as relations accessable by requestors who wish to know the description or structure of the data base.

The dictionary is the symbol table for the parse routines. All bindings are made at request execution time, except for string models to the physical data structure which is done at physical data base definition time. An optional link-time binding capability might be allowed for host-language calls. Temporary entries are created for any intermediate relations or variables defined over the scope of compound requests.

String models also reside in the Dictionary, where they are associated with conceptual objects they map to the internal model. Together with selection criteria (rules defining what subsets of data are found on an instance of a string), the models are input to the string resolution and code generation routines.

4 DRM SYSTEM ARCHITECTURE

This chapter describes the structure and control flow of the DRM data management system. As previously stated, DRM is intended to provide the benefits of a relational interface to users with a large current investment in conventional data base technology. The configuration, detailed herein, consists of a query and manipulation language accessing hierarchic structures managed at the physical level by IMS. This particular configuration was chosen for illustration. DRM is intended to access structures of various physical Data Base Management Systems.

Syntax of the principal user language statements is explained to promote understanding of the following examples of system internals. The reader will note the simplicity and power of the various operations. Features of the language are considered in context of system requirements with which they are aligned.

Internal functions of DRM are examined in the final section of the chapter. Module descriptions describe

the algorithms for their operation rather than code for their implementation.

4.1 USER LANGUAGE

The DRM stand-alone user language is called Data Resource Access FaciliTy or DRAFT. It is based upon SEQUEL 2 <6>. Statements provide the definition, access, manipulation, and control facilities to meet requirements defined in the previous chapter.

Data descriptions in the conceptual and external realms, as well as mappings between the two, are defined with DRAFT statements. Various external views are possible, with complexity limited only by the power of the SELECT statement.

The data control statements enable the DBA to grant different levels and extents of authority to different users. Assertions about data integrity and control functions are invoked by the system transparent to the requestor.

Examples in the following sub-sections on language syntax use the DEPARTMENT, EMPLOYEE, APPLICANT, and POSITION relations shown in figure 4.1.

Descriptions of the internals of DRM (section 4.3) use expressions in DRAFT for illustration.

4.1.1 QUERY STATEMENTS

The fundamental data accessing operation of DRAFT is the SELECT. It may appear as a clause in most other statements (see update for example) or as a stand-alone query statement. Selection extracts FROM a relation a subset of the tuples satisfying the conditions of the Where clause.

Q1 List the records of all programmers.

Selection may appear as an operation within a more complex inquiry.

Q2 What is the name of the manager of
department 23?

Projection is the choosing of a subset of the attributes from the tuples of a relation. The requestor

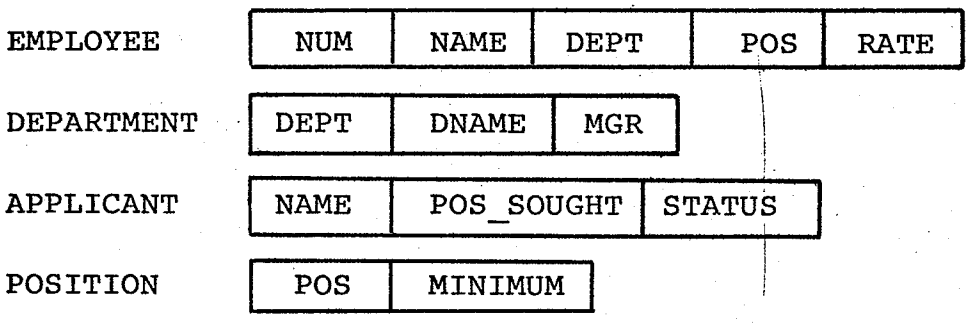


FIG. 4.1 RELATIONS FOR DRAFT EXAMPLES

```
Q1    SELECT   *   FROM  EMPLOYEE
      WHERE    POS = 'PROGRAMMER'

Q2    SELECT  NAME  FROM  EMPLOYEE
      WHERE   NUM = SELECT  MGR  FROM  DEPARTMENT
                    WHERE  DEPT = 23

Q3    SELECT  POS,NAME  FROM  EMPLOYEE

Q4A   SELECT  NAME,DNAME  FROM  EMPLOYEE,DEPARTMENT
      WHERE  EMPLOYEE.DEPT = DEPARTMENT.DEPT

Q4B   SELECT  NAME,DNAME  FROM EMPLOYEE X, DEPARTMENT Y
      WHERE  X.DEPT = Y.DEPT
```

specifies the projection by a list of attribute names in the SELECT (the names may be arranged in any order the requestor wishes).

Q3 List the position and name of all employees.

Join is the catenation of tuples from two relations satisfying a condition between attributes from each relation which share a common value set.

Q4 List all managers and the name of the department they manage.

The common relation here is DEPT. In format A the relation name has been used to resolve the ambiguity of the same name appearing in two relations. Format B illustrates the use of variable names to achieve the same qualification.

4.1.2 UPDATE STATEMENTS

The user may submit DRAFT statements to INSERT,

DELETE, or UPDATE instances in relations.

For insertions attributes to be initialized are specified by the INTO clause (optional with a default of all attributes). Initial values are specified by a a constant tuple:

U1 Insert a new employee with the given particulars.

or the results of a selection.

U2 Enter all hired applicants into the employee table.

Deletion removes instances of a given relation that satisfy the WHERE clause from the data base.

U3 Delete all applicants who have been hired

Update statements modify the values of attributes according to the expressions in the SET clause. New

```
U1    INSERT  INTO  EMPLOYEE:
            <504, 'BLOGGINS', 64, 'CLERK', 145.90>

U2    INSERT  INTO  EMPLOYEE:
            SELECT  NAME,POS_SOUGHT  FROM  APPLICANT
            WHERE   STATUS = 'HIRED'

U3    DELETE APPLICANT
      WHERE  STATUS = 'HIRED'

U4    UPDATE  EMPLOYEE E
      SET  RATE = SELECT MINIMUM FROM POSITION P
                  WHERE P.POS = E.POS
      WHERE  E.RATE = NULL


D1    CREATE  TABLE  EXPERIENCE
                  (NAME (CHAR(20)),
                   POS  (CHAR(10)),
                   YEARS (INTEGER))

D2    DEFINE  VIEW  EMPDEPT  AS:
            SELECT  NAME,DNAME
            FROM EMPLOYEE E, DEPARTMENT D
            WHERE  E.DEPT = D.DEPT

D3    DROP  VIEW  EMPDEPT
```

values may be stated as constants, expressions, or SELECT clauses.

U4 Set any employees with no salary rate
to the minimum for their position.

4.1.3 DATA DEFINITION STATEMENTS

Data definition statements of DRAFT allow the DBA to CREATE relations, DEFINE views, and DROP either data definition or control declaratives from the Dictionary.

The CREATE statement enters a new TABLE (DRAFT name for a relation), along with its FIELD (attribute) definitions, into the Dictionary.

D1 Define an experience relation with
20 character name, 10 character position,
and binary years attributes.

External views of relations are created via the DEFINE statement. This example is the first appearance

of a selection in a declarative. While many systems severely restrict external view (usually to projections from conceptual records), DRM allows the full power of its query facility to be used in defining views. D2 uses selection, join, and projection.

D2 Define a view containing all employee
names with their department name.

System entities, such as assertions, tables, and views, are deleted from the Dictionary with a DROP.

D3 Delete the EMPDEPT view from the system.

4.1.4 CONTROL STATEMENTS

Protection of the security and integrity of the data base are critical to successful management of data. Control facilities of DRAFT consist of statements to GRANT or REVOKE various degrees of authority to users, ASSERT validity conditions for modification of values in the data base, automatically TRIGGER monitoring or

```
C1    GRANT  READ  ON  POSITION,
             READ, UPDATE(POS,RATE)  ON  EMPLOYEE
      TO  FRANK,DICK


C2    REVOKE  INSERT  ON APPLICANT
      FROM    DON


C3    ASSERT PROG_DEPT  ON  EMPLOYEE X:
             IF POS = 'PROGRAMMER'
             THEN 'SYSTEMS' = SELECT  DNAME
                              FROM  DEPARTMENT Y
                              WHERE  Y.DEPT = X.DEPT


C4    DEFINE  TRIGGER  NEWEMP
      ON  INSERT  OF  EMPLOYEE E:
          DELETE APPLICANT A
          WHERE  A.NAME = E.NAME
```

integrity preserving operations when certain conditions occur.

The GRANT statement specifies relations a user may "see" and what operations he may perform upon them. Authority is exercisable down to the attribute level (note the update option in example C1).

C1 Authorize users Frank and Dick to read positions and employees and update the position and rate attributes of employees.

Some or all of a user's privileges may be rescinded by means of a REVOKE statement.

C2 Revoke Don's authority to insert applicants.

Any combination of GRANT and REVOKE statements may be used to define or modify a user's profile.

The DBA may ASSERT conditions that must be satisfied before an update operation is allowed. Assertions (they might be thought as system edits) are used to

implement installation defined integrity controls over the values in the data base. If an insert, update, or delete operation violates an assertion, the entire request is terminated and an abnormal condition is returned.

C3 No employee may have the position of programmer unless assigned to the Systems Department.

Often the DBA will want to assure that some action occurs if a particular operation is executed against the data base. DRM will automatically invoke actions defined by a TRIGGER declarative whenever the specified ON condition occurs.

C4 Delete a person's application if that person is added as an employee.

4.2 SYSTEM INTERNALS AND CONTROL FLOW

An initial implementation of DRM will consist of a stand-alone language interface to access data bases stored in a conventional data base organization. The system flow, described in this section, is typical of such an implementation.

Techniques of structured design <45> were used to decompose the system into modules. Figure 4.2 is known as a structure chart with boxes for modules, solid lines for control flow, and named arrows to indicate data flows.

On the left, or input, leg DRAFT statements stated in terms of the external data structure are translated into optimized storage representation dependent code. In the centre, or transform, leg the internal code is interpreted. Output data and messages are returned to the requestor via the right, or output, leg.

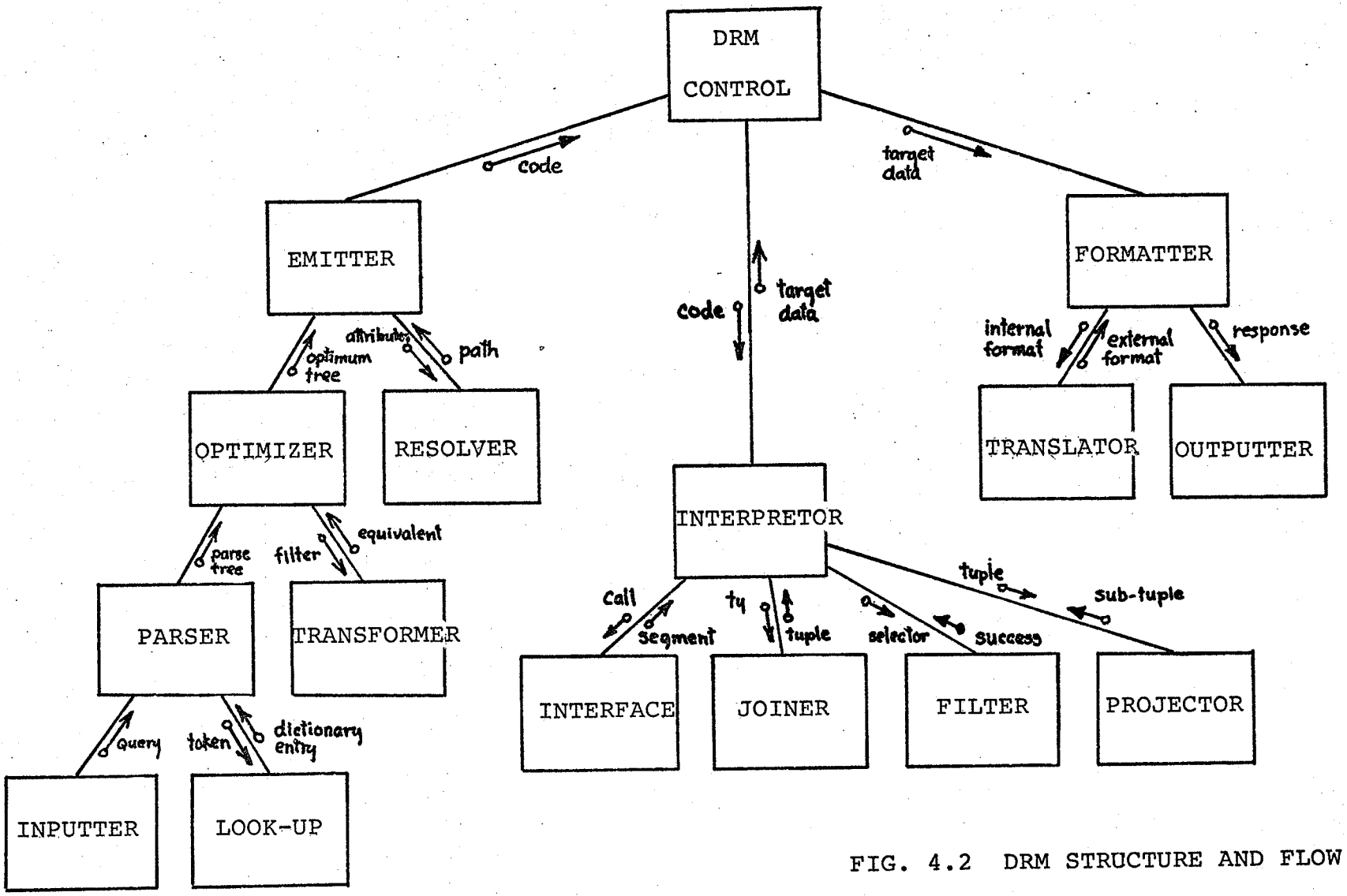


FIG. 4.2 DRM STRUCTURE AND FLOW

4.2.1 PARSE REQUESTS

The DRAFT parser translates requests from the external language into the internal representation referred to as the parse tree. Names are replaced by their symbol table entries and view defining expressions (if appropriate). Assertions and triggers on the requested actions are introduced into the parse tree during this phase. In parallel with translation, the request is tested against the privileges specified by previous GRANT statements. A violation of authority will cause termination of the parse and triggering of the system security monitoring functions.

While expressions are actually kept in an efficient but unreadable internal form, they will be shown in a graphic form for illustration. Additional operators to the normal arithmetic and boolean are shown in Figure 4.3. Figure 4.4 is a typical selection expression tree in grphic form. Manipulation operations may consist of up to three steps: input (selection), process (value setting, etc.), and output (insert, replace, delete). A typical example is shown in Figure 4.5.

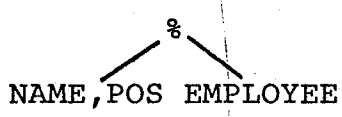
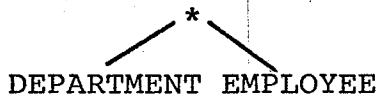
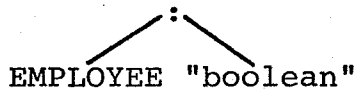
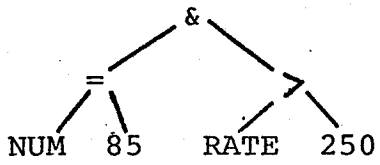
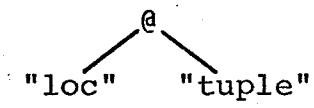
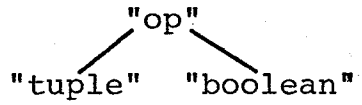
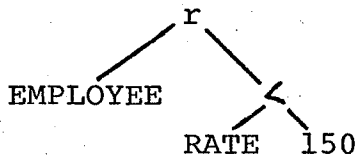
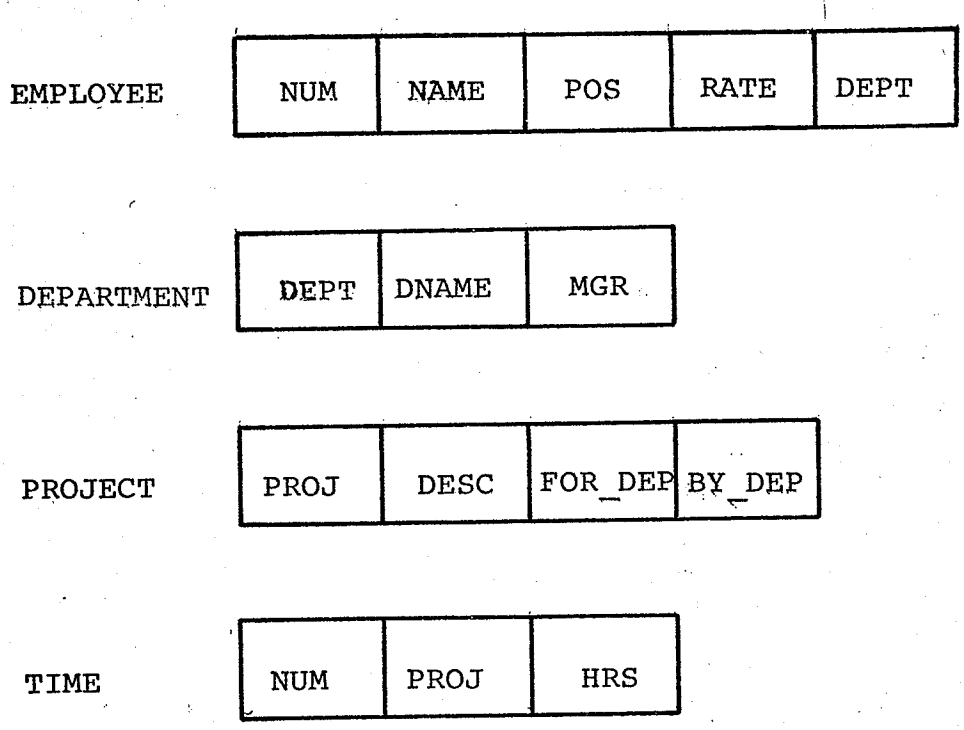
OPERATOR	NAME	EXAMPLE
%	PROJECT	
*	JOIN	
:	SELECT	
	"boolean"	
@	ASSIGN	
i	INSERT	ALL OF THE FORM:
r	REPLACE	
d	DELETE	

FIG. 4.3 DRAFT INTERNAL OPERATORS




```
DEFINE  VIEW  PROJ_LIST  AS:
        SELECT    *    FROM  DEPARTMENT,PROJECT
        WHERE  DEPT = FOR_DEPT


DEFINE  VIEW  DEPT9  AS:
        SELECT  NUM,NAME,POS,RATE
        FROM  EMPLOYEE
        WHERE DEPT = 9
```

FIG. 4.4 RELATIONS FOR INTERNALS EXAMPLES

```
SELECT   NAME,POS,MGR
FROM   EMPLOYEE E, DEPARTMENT D
WHERE   NUM = 608 & E.DEPT = D.DEPT
```

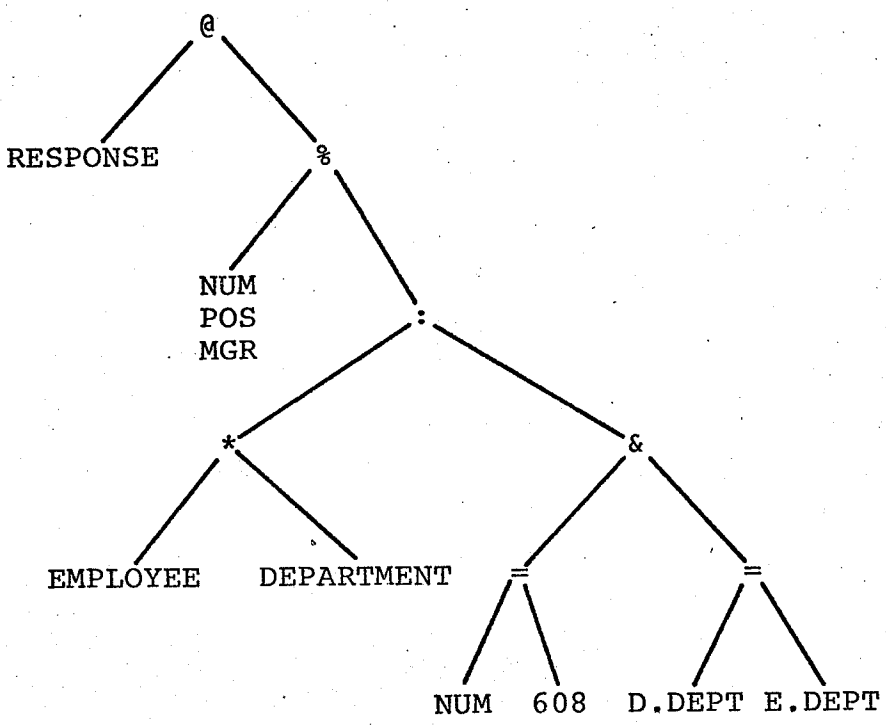


FIG. 4.5-A TYPICAL SELECT PARSE TREE

```
UPDATE   EMPLOYEE:
         SET   RATE = RATE x 1.06
WHERE    NUM = 269
```

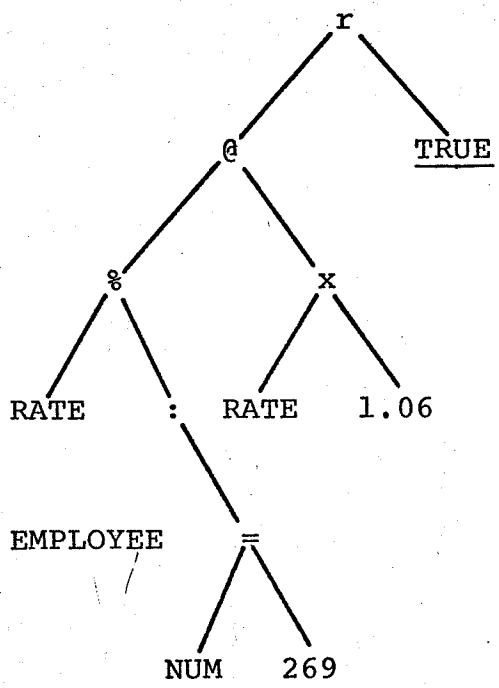


FIG. 4.5-B TYPICAL UPDATE PARSE TREE

The parser is of the LALR(k) type <17>. To maintain flexibility to enhance the language parser, decision rules (state tables) are created directly from the BNF description of the syntax by a generator program <29>. It is possible to create some meaningless constructs, since a few ambiguities have been allowed in the syntax to preserve understandability. Parser routines must include the ability to recover from these situations.

Symbol table look-up is the function of replacing external names with equivalent dictionary entries. Since views are defined as selection expressions, the replacement includes closure or the substitution of external views by expressions entirely in terms of objects in the conceptual data structure (Fig 4.6).

Assertions about the data accessed by the request and actions triggered by the resulting operations are introduced into the parse tree during translation. For example, the control defined in Figure 4.7 would modify the parse of Figure 4.5 as shown.

Requested actions are tested against the requestors

```
SELECT    *    FROM  DEPT9
WHERE   POS = 'ANALYST'
```

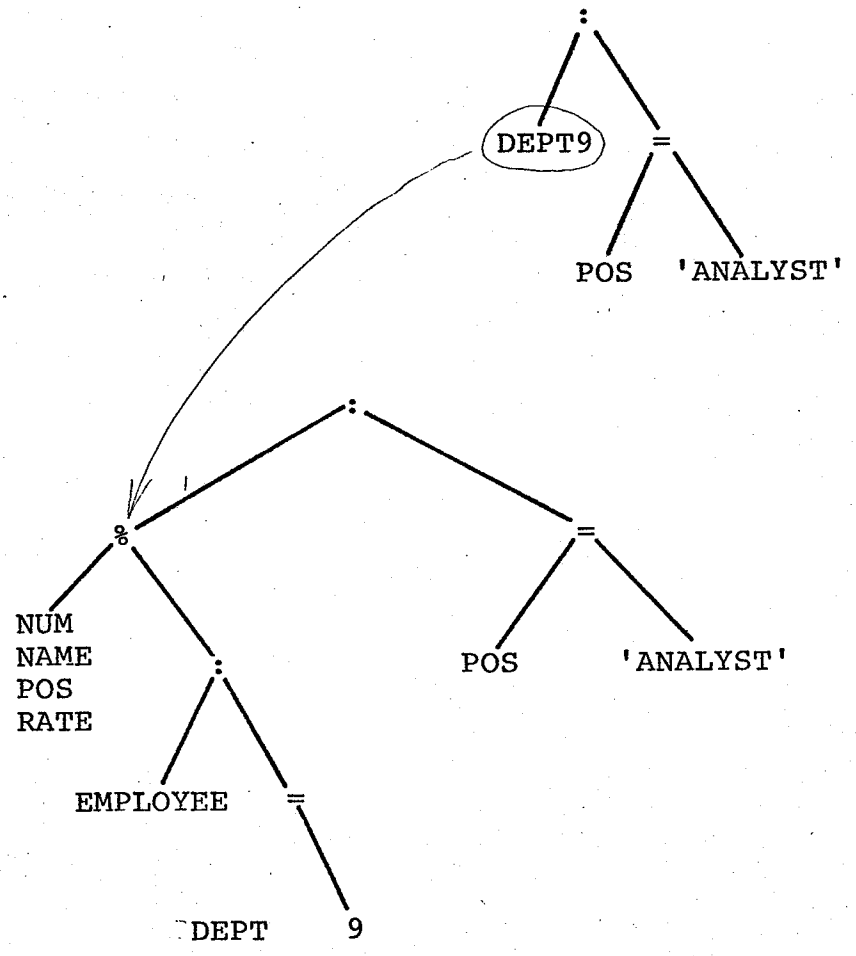


FIG. 4.6 CLOSURE WITH VIEWS

Consider the update from Fig. 4.5-B with:

ASSERT MAX_RATE ON EMPLOYEE:

RATE ≤ 300

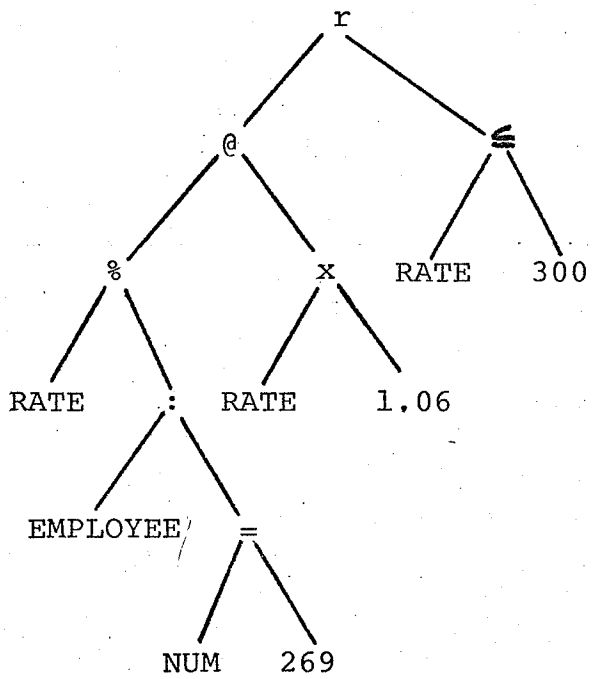


FIG. 4.7 ASSERTION ON UPDATE

privileges during translation. Authority given by GRANT statements are applicable to object types rather than particular instances and, therefore, may be fully verified without reference to the actual target data. Granting at the type level is no severe limitation as authority can be further restricted to subsets of the object instances by VIEW defining statements.

4.2.2 OPTIMIZE SELECT CLAUSES

Optimization is the process of transforming the structure and order of the request to improve its performance, while preserving its intent. In an environment where the internal storage structure is hidden from the requestor, queries will not always be stated in an optimal form. The system, with knowledge of the current structure, reprograms the request into a form more amenable to efficient execution.

The algorithms described in this section do not guarantee an optimum form, but do take steps that have a high probability of improving performance. Experi-

ments conducted by Hall <25> showed some improvement in most cases, marked improvement with "unreasonable" (very poorly stated in terms of the physical structure) requests, and only a single case of de-optimization.

Three phases are implemented in Hall's optimizer. The last or maxi phase replaces common sub-expressions with secondary relations. It was decided not to attempt this optimization since the interpreter pipe-lines all operations (the whole expression is performed a tuple at a time). Instead the methods are similar to those of Smith and Chang <41>,which parallel those of Hall but for one exception. They attempt to execute project operations as early as possible, while Hall only recognizes the possibility of such a transform. Algorithms used in DRM do not attempt to move projections, but the emitter causes the equivalent through a minimum query covering by mapping only attributes required for selection (qualification criteria) and output.

The order of execution of the transform functions minimizes reiteration in optimizing a given expression. In order the transforms are:

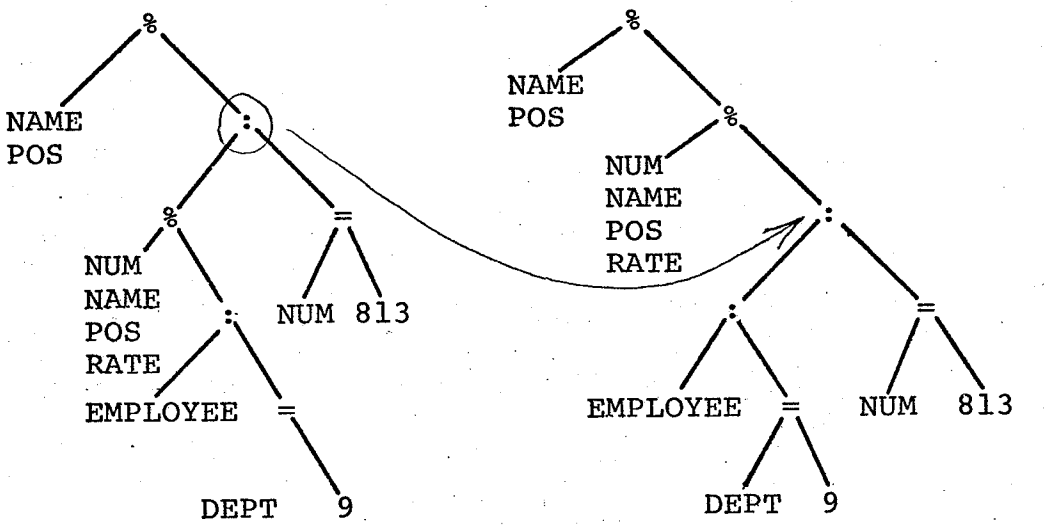
- Combine sequences of selections into a compound selection.
- Move selections down the parse tree.
- Distribute selects through joins.
- Combine sequences of projections.

Transformations apply set identities and equivalences to preserve the intent of the original request.

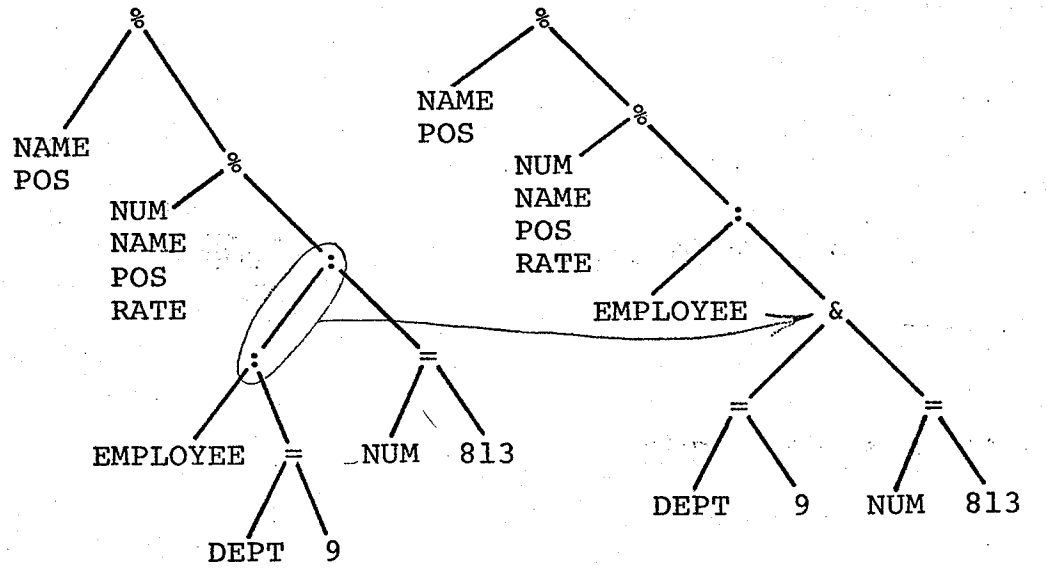
Sequences of selections are combined by replacing them with a single select consisting of the conjunction of the original selections. The resulting compound selection can be moved down the tree past other nodes with less effort than reiterating the move for each component. Figure 4.8-A illustrates a combination of selections.

By performing selection as early as possible, the number of tuples processed by further operation is reduced. Selections which reach the leaves of the parse tree - the interface with stored data - refine the resolution of the most favorable access path since paths are defined in terms of subsets of data that they select. Partitioning of a request by strings tends to

```
SELECT   NAME,POS   FROM   DEPT9
WHERE    NUM = 813
```



A. MOVE SELECTIONS



B. COMBINE SELECTIONS

FIG. 4.8

reduce the number of physical operations to complete the request. (Note the importance of minimizing I/O since the cost of secondary storage access is much greater than that of primary storage access.) Figure 4.8-B shows the results of a selection move transform.

In general, the factors of a selection do not apply uniformly to both relations participating in a joining operation. To distribute the select past a join (say A*B) it is necessary to decompose the boolean filter, S, into an equivalent conjunction Sa & Sb & Sr, where Sa refers only to attributes of A, Sb only to those of B, and Sr to attributes of both relations. Selectors Sa and Sb are then distributed past the join down their respective sub-trees. Figure 4.9 illustrates distributing a filter through a join.

Combining sequences of projections can improve performance by eliminating redundant intermediate move operations. Consider how the project operator works:

PROJECT(<I1,I2,...In>,<A1,A2,...Am>)=

<Ai1,Ai2,...Ain>

```
SELECT   PROJ,DNAME   FROM   PROJ_LIST
WHERE    DESC = 'REPAIR'
```

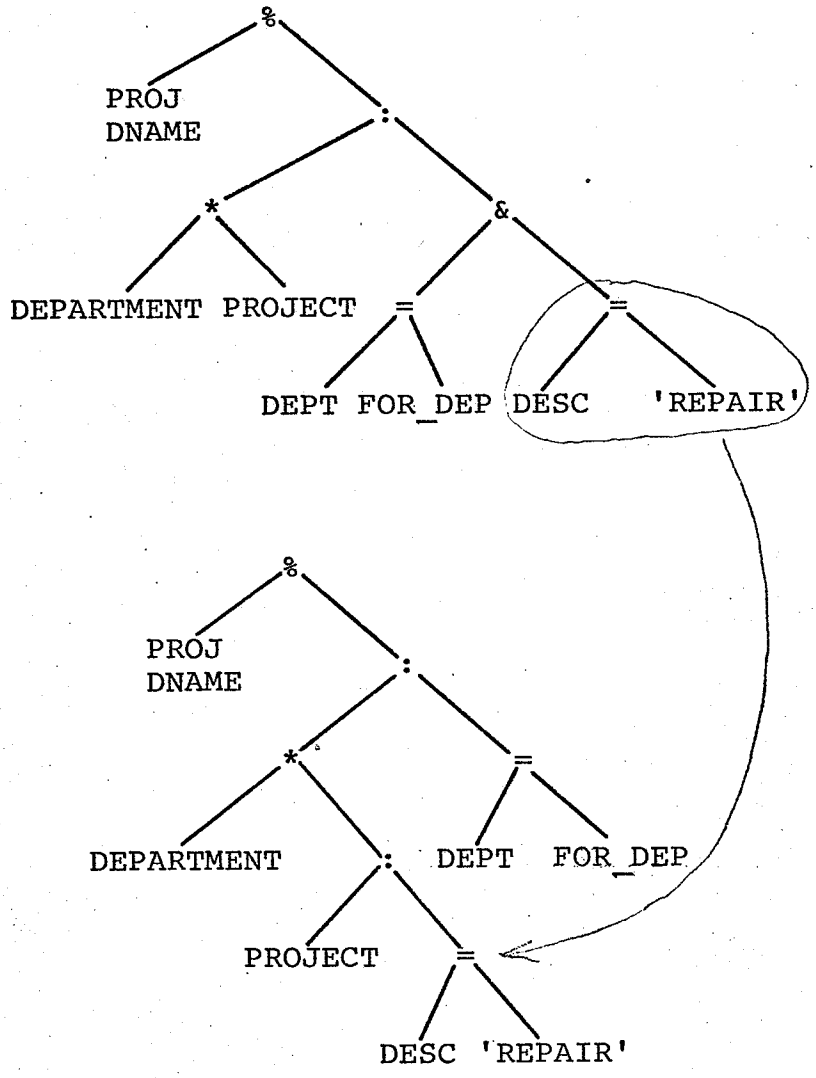


FIG. 4.9 MOVE FILTER PAST JOIN

ie PROJECT(<1,4,3>,<A,B,C,D>)=<A,C,D>

PROJECT(<3,2>,<A,C,D>)=<C,D>

PROJECT(<2>,<C,D>)=<D>

resulting in six moves, could be replaced by

PROJECT(<4>,<A,B,C,D>)=<d> with one move.

Figure 4.10 illustrates the combination of projections.

4.2.3 RESOLVE PHYSICAL ACCESS PATHS

Resolution is the function of finding paths through the physical data base which access all the instances of data satisfying the query. In this module the query is characterized by the attributes to be output or that appear in the conditions which define the subsets of data to be fetched. These parameters are defined at the conceptual level. The string model specifies the mappings of these attributes into the external level.

The resolver searches for all paths satisfying the

Consider the result of Fig. 4.8-B:

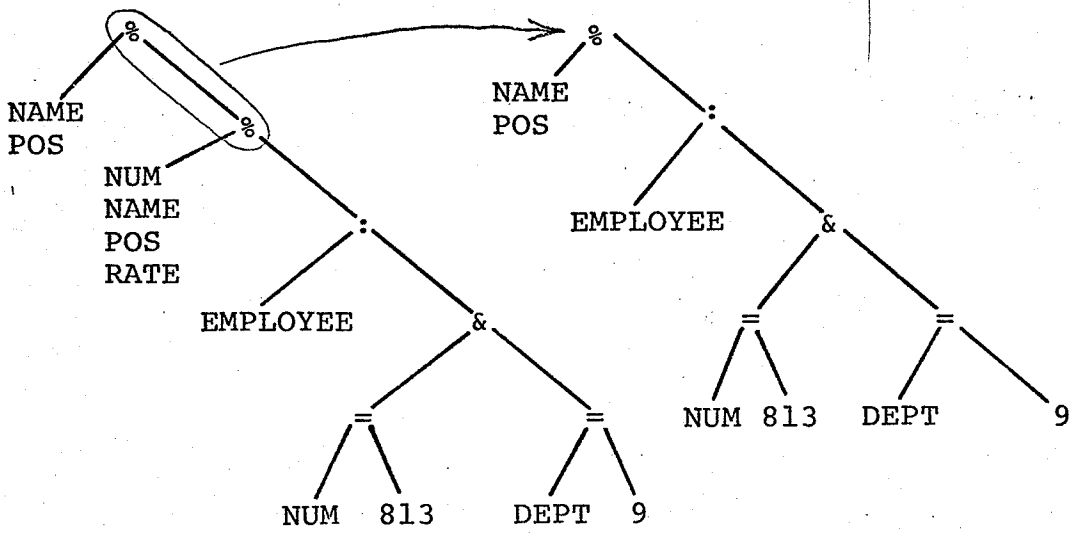


FIG. 4.10 COMBINE PROJECTIONS

query. Should more than one path exist, the most favorable path is selected. What is "satisfactory" and "most favorable" will be considered at some length.

Algorithms used in this module are based on the work of Ghosh and Senko <22>. The query is parameterized by specification of the qualification criteria (QC), which are the selections that have migrated to the leaves of the parse tree, and the output attributes (OA). While an attribute may serve a dual role, the OA will mean those which are not also in the qualification (QCA).

The search for a path to the physical data, which satisfy the query, begins with locating the QCA and OA. A feasible string is one that:

- covers all the QCA and OA
- connects at least all the relevant instances of the attributes
- terminates at an entry point to the physical data network.

If more than one feasible path exists, the most favorable path will be the one with the minimum path cardinality (PC). For this thesis, PC is defined as

the sum of the number of instances of all strings (A,E, or L) likely to be connected by the access path. The PC of an E or L-string is based upon its frequency function (the expected number of instances of strings on the exit list per instance of a string). Thus PC is the most probable length rather than an exact value. No attempt is made to calculate the exact length, since the effort would not justify the cost (likely following the physical path).

The methodology is first described for a single entity. The resolver begins by identifying the A-string(s) which cover the QCA and OA. Each string covering the string(s) is examined to see if its SC covers the query. Any string whose SC negates the QC (that is, which selects a subset of the data disjoint from the query) is rejected. If more than one covering string exists, the one with minimum PC is selected.

At each step of the process, should no covering string be found, a set of strings, which share a common ID attribute or ID-conjugate (L-string), covering the query is sought. If none exists the query cannot be

answered by the data base. Otherwise, the QC is decomposed into subsets related to each string in the cover, and the process repeated for each subset.

As the search for entry points is carried out, a record of the PC of each path is maintained. If more than one feasible path is found, the path with least PC is selected. Should there be more than one with the minimum PC, selection is based on the relative clustering of the strings (ie in IMS: with a "physical" and a "logical" path of equal PC, the physical path would be chosen because of its proximity to physical ordering).

When a request involves more than one entity, the QC is decomposed into subsets pertaining to the entities individually. The single entity search process is applied to each entity and related QC subset. If any L-string connecting relevant entities is encountered, the QC is modified accordingly and the search resumed.

The conjunction of the SC of the various strings forming a path (called the path criteria or PC) defines the subset of the data base selected by the path. For

```
SELECT   NAME,POS   FROM EMPLOYEE
WHERE    DEPT = 5
```

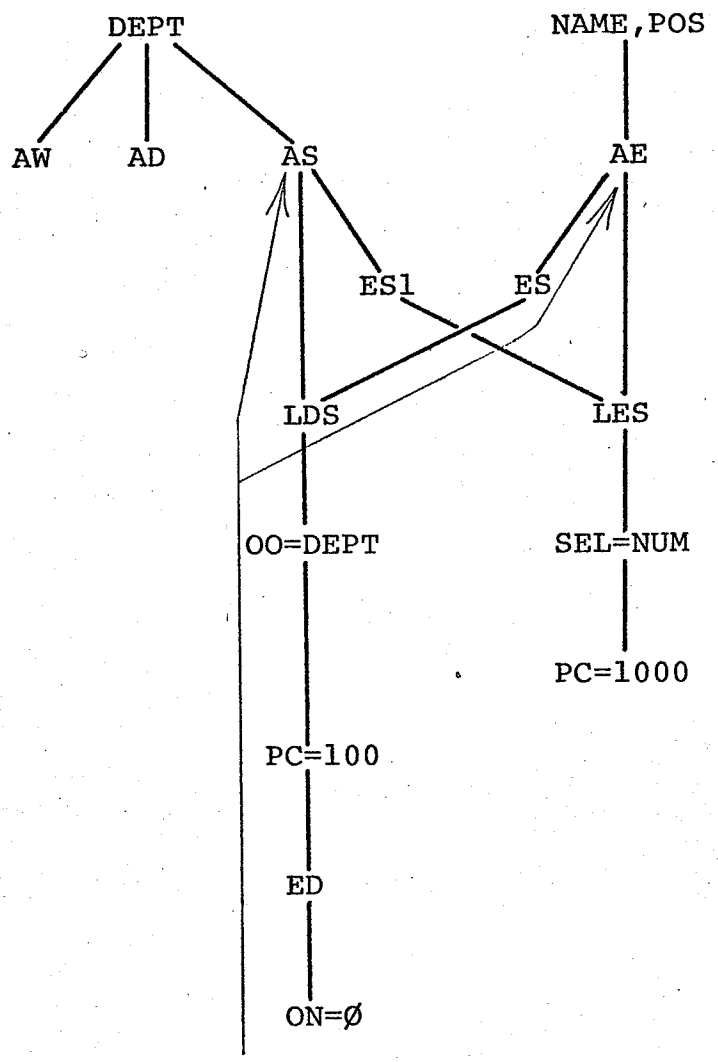


FIG. 4.11 SAMPLE RESOLUTION

a selected path that subset may either be identical to or contain the target data. The subsets are identical if and only if the PC is identical to the Qc. Otherwise, operations will be required to select instances of the target set from the path set. An analytical expression for the difference QC-PC is needed by the emitter to generate the instructions to perform this Residual Selection (RS).

Figure 4.11 shows a sample resolution graphically.

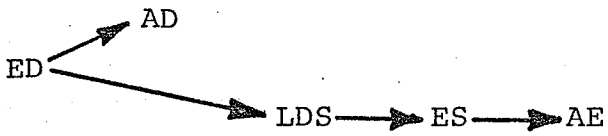
4.2.4 EMIT CODE

The emitter generates, from the parse tree, the sequence of instructions to be invoked by the interpreter to answer the request. various set operations are imbedded in the looping, data base interface, and status checking instructions required to follow the physical path chosen by the resolver. Finally, instructions are generated to return the final results and status to the controller for output to the requestor.

Join operations present the most difficulty to the emitting function. In the most general case, join is equivalent to the cartesian product (that is, all possible pairs consisting of one tuple from the first relation followed by a tuple from the second relation). Most often the requestor only wants those pairs which satisfy a boolean condition between the tuples (and most often of these: a match on two attributes with a common set of values). Significant improvements in system performance are gained by implementing the join and selection as a compound operation. As well as this combination the emitter employs several strategies <23, 40> to optimize joining by taking advantage of:

- indices
- relative cardinality
- L-strings which cover the join

To determine the data base calling sequence selected paths are analyzed beginning from the entry points. For each E-string encountered a call is generated. The qualification clause of the call is determined by the portion of the residual selection (RS) that is resolvable at this level. The RS is decomposed by




```
GET  DEPSEG (DEPT=5);                    /* AD ON ED */

GET VIA LDS  EMPSEG;                     /* AE ON ES */

IF NOT FOUND(EMPSEG) THEN SET(EMPTY);

WHILE FOUND(EMPSEG);

     T = PROJECT((NAME,POS), EMPSEG);

     ASSIGN(T, RESPONSE);

     GET WITHIN LDS  EMPSEG;        /* NEXT AE */

END;
```

FIG. 4.12 SAMPLE EMIT

deletion of the satisfied qualifiers and code generation is resumed. All subsequent calls are restricted to be WITHIN the current position of all higher levels on the path (ie GNP call in IMS). Loops are generated to exhaust instances of each E-string on an instance of its superior string. Status checking instructions test the existence and exhaustion of subordinate instances.

The query and selected path from Figure 4.11 results in the emitted code in Figure 4.12.

4.2.5 INTERPRET REQUEST CODE

The interpreter "executes" the emitted sequence of instructions by calling the various service functions, simulating the looping, and monitoring the status returned by the sub-functions.

The primary service function is the data base interface module. Data base call requests are received and translated into equivalent calls in the format of the

physical DBMS (in our case DL/1 calls for IMS). The interface locates the necessary control blocks and translates DBMS status codes to interpreter return codes.

Other modules are provided to PROJECT attributes from tuples, enter tuples in the result, allocate and maintain buffers, and miscellaneous services.

5 SUMMARY

In this thesis, we have looked at several trends in the management and technology of data base systems. The concept of a pool of all enterprise data, capable of answering all possible questions is not feasible. Instead many today see data as a resource, limited in it's extent, that should be organized to maximize its value in a business sense. A set of objectives were adopted to support this data resource approach.

It was shown that, in the short term, the need for management information can best be served by provision of a very high-level data language accessing typical conventional data base organizations. To preserve the option to migrate to new technologies as the opportunity arises, independence of the language (and hence the user) from the physical structure must be attained.

An examination was made of the Relational Model of data, a logical data organization whose conceptual simplicity allows a greater degree of data independence than the conventional hierarchic or network models.

Additionally, the relational approach has the sound mathematical basis to guarantee its completeness and power. Although well developed academically, no commercial relational system is available today. In spite of this, many people (the author included) believe the relational model will become the dominant approach to data base in the next decade.

The DRM architecture was developed as a vehicle to prove the feasibility of a relational external model covering a conventional internal system. Through the three level structure and string modelling, the system also has the flexibility to migrate to new storage technologies without undue hardship to the DBA, and the independence to do so without impacting current users. DRM has demonstrated this feasibility with sufficient function to meet all of the GUIDE/SHARE requirements for data management systems.

To maintain independence and yet reasonable performance, DRM optimizes requests in much the same way a good programmer would construct requests if the actual data structure were known. The requestor is free to

state requests without concern for the "goodness" of the formulation, content that the system will transform the request to an equivalent form which is amenable to efficient execution.

The system was designed to operate a tuple at a time. This approach is closely with the segment per call and path following characteristics of the example DBMS: IMS. The alternative of a relation at a time could have been used without change to the external interface. A detailed comparison should be made in the future to determine which approach is best. Further study should also be given to the possibility of saving target relations between requests, thus improving the performance of related requests.

Algorithms for DRM modules and their interfaces were described. Their compatibility was proven empirically by translation of a series of representative requests. Format and methodology are meant to suggest rather than define the implementation. Slight modifications will be desirable (and sometimes necessary) to accomodate the practicalities of a real system. It has been left

for the implementor to select the best representation of the parse, internal lists, etc. In particular, more study of the organization of the Dictionary is needed.

An example of a first phase DRM system was given using the DRAFT language (a variant of the existing SEQUEL 2) and IMS as the physical data base management system. The approach is equally applicable to other hierarchic or network structures. Several stand-alone requests were carried from input through interpreter to illustrate the operation of the DRM modules.

DRAFT also provides host language coupling, allowing development of programs in PL/I or COBOL, which will be independent of physical structure changes. A topic for future study is the feasibility of DRM providing interfaces from programs which issue calls to conventional systems (IMS's DL/1 or DBTG for instance) to newer storage technologies as they are implemented.

BIBLIOGRAPHY

Abbreviations:

CACM = COMMUNICATIONS OF THE ACM

IBM J of R & D = IBM JOURNAL OF RESEARCH AND DEVELOPMENT

1 Anthony,R.N., PLANNING AND CONTROL SYSTEMS: A FRAMEWORK FOR ANALYSIS, Harvard School of Business, Boston, 1965

2 ANSI/SPARC Study Group on DBMS, "Interim Report", ANSI, Washington, Aug. 1975

3 Astrahan,M.M., et al, "SYSTEM R: A Relational Approach to Data-Base Management", IBM Res Rep RJ1738, Feb. 1976

4 Casey,R.C., & Delobel,C., "Decomposition of a Data Base and the Theory of Boolean Switching Functions", IBM J of R & d, Sept. 1973

5 Chamberlin,D.D., Gray,J.N., & Traiger,I.L., "Views,

Authorization, and Locking in a Relational Data Base System", DATABASE MANAGEMENT SYSTEMS, AFIPS Press, Montvale, NJ, 1976

6 _____, et al, "SEQUEL 2: A Unified Approach to Data Definition, Access, and Control", IBM J of R & D, Nov. 1976

7 Childs, D.L., "Feasibility of a Set-Theoretic Data Structure - A General Structure Based on a Reconstituted Definition of a Relation", PROC. 1968 FJCC, AFIPS Press, Montvale, NJ, 1968

8 CODASYL Data Base Task Group, "April 1971 Report", ACM, New York, NY, 1971

9 CODASYL Development Committee, "An Information Algebra", CACM, New York, NY, Apr. 1962

10 Codd,E.F., "A Relational Model of Data for Large Data Banks", CACM, New York, NY, JunE 1970

11 _____, "Relational Completeness of Data-Base Sublanguages", COURANT COMPUTER SCIENCE SYMPOSIUM 6: DATA BASE SYSTEMS, Prentice Hall, Englewood Cliffs, NJ, 1971

12 _____, "Further Normalizations of the Data Base Relational Model", ibid.

13 Cullinane Corp., INTEGRATED DATA MANAGEMENT SYSTEM Wellesley, MA

14 Date,C.J., "An Architecture for High-Level Language Constructs for Data Base", PROC GUIDE 44, GUIDE International, Chicago, Il, 1977

15 Deardon,J., "MIS Is a Mirage", HARVARD BUSINESS REVIEW, Boston, MA, 1972

16 Dennis,J.B. & Van Horn,E.C., "Programming Semantics for Multiprogrammed Computations", CACM, New York, NY, March 1966

17 DeRemer,F., "Simple LR(k) Grammers", U. of Cal. at

Santa Cruz, 1971

18 Earley,J., "Towards an Understanding of Data Structures", CACM, New York, NY, Oct. 1971

19 Engles,R.W., "A Tutorial on Data Base", IBM Tech Rep TR00-2004, 1970

20 Eswaran,K.P., et al., "On the Notions of Consistency and Predicate Locks in a Data Base System" IBM Res Rep RJ1487, Dec. 1974

21 _____, & Chamberlin,D.D., "Functional Specification of a Sub-system for Database Integrity", PROC. of the INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, Framingham, MA, Sept. 1975

22 Ghosh,S.P., & Senko,M.E., "String Path Search Procedures for Data Base Systems", IBM J of R & D, Sept. 1974

23 Gottlieb,L.R., "Computing Joins of Relations", PROC.

ACM-SIGMOD CONF., ACM, New York, NY, 1975

24 GUIDE/SHARE, "Joint Committee Report on the Requirements for DBMS", GUIDE International, Chicago, IL, 1970

25 Hall,P.A.V., "Optimization of Single Expressions in a Relational Data Base System", IBM J of R & D, May 1976

26 Hammer,M.M., & McLeod,D.J., "Semantic Integrity in a Relational Data Base System", PROC. of the INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, Framingham, MA, 1975

27 Held,G., & Stonebraker,M., "Storage Structures and Access Methods in the Relational Data Base Management System INGRES", PROC. ACM PACIFIC REGIONAL CONFERENCE 1975, ACM, New York, NY, 1975

28 IBM Corp., INFORMATION MANAGEMENT SYSTEM / VS, New York, NY

29 Lalonde,W.R., Lee,E.S., & Hornung,J.J., "An LALR(k) Parser Generator", PROC. IFIP CONGRESS 1971, North-Holland Publ., Amsterdam,1972

30 Lorie,R.A., "XRM - An Extended(n-ary) Relational Memory", IBM Scientific Centre Report G320-2096, Cambridge, MA, Jan. 1974

31 Mairet,C. "Data Structures: The GUIDE/SHARE Approach to Change", PROC. of the DPI CONF. 1975, Ottawa, Ont., 1975

32 Martin,J., SECURITY ACCURACY AND PRIVACY IN COMPUTER SYSTEMS, Prentice-Hall, Englewood Cliffs, NJ, 1976

33 _____, COMPUTER DATA-BASE ORGANIZATION, Prentice-Hall, Englewood Cliffs, NJ, 1975

34 MRI Systems Corp., SYSTEM-2000, Austin, TX

35 Myopoulos,J., Schuster,S., & Tsichritzis,D., "A Multi-Level Relational System",

PROC. NATIONAL COMPUTER CONFERENCE 1975,
AFIPS Press, Montvale, NJ, 1975

36 Nolan,R.L., "Computer Data Bases: the Future is Now", HARVARD BUSINESS REVIEW, Boston, MA, Sept. 1973

37 _____, MANAGING THE DATA RESOURCE FUNCTION, West Publishing, Boston, MA, 1974

38 Reece,J.W., "Normalization as a Tool for DL/I Data Base Design", PROC. CANSAC/BIS CONFERENCE, Toronto, Ont., Feb. 1976

39 Senko,M.E., et al., "Data Structures and Accessing in Data Base Systems", IBM SYSTEMS J., Vol. 12 Num. 1, p 30, 1973

40 Seth,R., "Smooth Roads to Systems That Yield Profits", MANAGEMENT CONTROLS, Peat Marwick Mitchell, New York, NY, Nov. 1976

41 Smith,J.M.,& Chang,P.Y., "Optimizing the Performance

of a Relational Algebra Database Interface", CACM, New York, NY, Oct.1975

42 Smith,L., "Data Independence", AUERBACH COMPUTER TECHNOLOGY REPORTS: DATABASE,Auerbach Publishing, Philadelphia, PA, 1976

43 Taylor,R.W., "Generalized DBMS Data Structures and Their Mapping to Physical Storage", PhD. Thesis, University of Michigan, Ann Arbor, MI, 1971

44 Wang,C.P., & Wedekind,H.H., "Segment Synthesis in Logical Data Base Design", IBM J of R & D, Jan. 1975

45 Yourdon,E., & Constantine,L.L., STRUCTURED DESIGN, Yourdon, Inc.,New York, NY, 1975

46 Zloof,M., "QUERY-BY-EXAMPLE: the Invocation and Definition of Tables and Forms", IBM Res Rep RC5115, 1975