A DATA ADMINISTRATION SUPPORT SYSTEM

Adolph D'Cunha

A Major Report

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

for the degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

December, 1982

ABSTRACT

A DATA ADMINISTRATION SUPPORT SYSTEM

Adolph D'Cunha

Data is gaining recognition as a valuable corporate resource. More and more business organizations are recognizing that the data resource must be managed the same way as traditional resources such as land, capital, labour, etc. Towards this end, creation of a 'Data Administration' function to administer data corporation-wide, is essential. The scope and dimension of the Data Administration undertaking is such that proper automated and manual tools are an absolute necessity. Data Administration Support System (DASS) proposed in this report is a computer aided facility that attempts to collect under one roof, the tools necessary for the administration of data. The work described here provides a generalized structure for a data administration support system and describes the implementation carried out for one of the Data Administration activities, namely, Conceptual Schema Design.

ACKNOWLEDGEMENTS

I am grateful to my supervisor, DR. T. Radhakrishnan for his assistence, advice and encouragement througout the DASS project.

I also wish to thank Dr. P. Goyal for reviewing this report and providing me with valuable comments.

TABLE OF CONTENTS

LIST OF FIGURES, TABLES AND APPENDICES

FIGURES

TABLES

APPENDICES

CHAPTER 1

DATA ADMINISTRATION - A NEW CORPORATE FUNCTION

1.1 DATA ADMINISTRATION - INTRODUCTION

1.1.1 DATA - A CORPORATE RESOURCE

A new attitude, which recognizes data as a resource, has been emerging in the 1980s. More and more business organizations are beginning to realize that:

- o data is a corporate resource just like other more traditional resources such as land, capital, labour, etc.
- o data is not free and has a value
- o data is an investment
- o data is needed to manage other resources
- o data is a shared resource
- o data should be managed as are other shared resources

The recognition of data as a resource creates a fundamental need for a centralized function much like property management, financial management, personnel management, etc. Of late, this centralized function has come to be known as Data Administration (DA) or Data Resource Management (DRM). We can view Data Administration function as a corporate endeavour committed to recognize

data as a resource and manage it effectively through appropriate policies, organizational infrastructure, tools/techniques, and controls.

It is believed that data resource management will ultimately be just one aspect of a broader area: information resource management (IRM). The latter concept includes all forms of corporate data such as voice data, image data, text data, etc. [Gillenson,1982].

1.2 DATA ADMINISTRATION - HISTORY OF EVOLUTION

The data administration function in most business organizations is still in an infantile stage. The lack of progress in the data administration field is attributable to the evolution of business computing without an overall plan. The computing technology has evolved so rapidly in the past twenty or so years that the transition for most large business organizations, which have a high degree of inertia, has not been an orderly one. The early business computing applications were ad hoc implementations to solve specific operational problems. In such cases, data sharing, data management and data control philosophies were absent. Consequently, when newer and more complex and functionally autonomous applications proliferated the scene, they were governed by 'entropy', i.e., the natural tendency in systems to move towards increasing disorder in an uncontrolled environment. The message was clear and precise: 'systems

intergration and data sharing'. The 1970's thus witnessed the promotion of a new concept 'the data base approach' and paved the way for DBMS - Data Base Management Systems [Flynn,1974; Palmer,1975a,b; Martin,1975; Fry,1976; Sibley,1976; Chamberlin,1976a; Taylor,1976; Tsichritzis,1976; Michaels,1976; Date,1977; Appleton,1977; Nijssen,1978; Cardenas,1979; Codd,1982a]. The database approach is based on three basic concepts:

- o data shareability
- o data non-redundancy
- o data independence

A database is a logically organized and structured collection of interrelated data stored together without unnecessary redundancy to serve multiple applications and diverse business information requirements. With DBMS implementation it became possible to integrate several related application systems, thereby achieving data shareability, non-redundancy and independence. The introduction of 'data dictionary' concept also facilitated ensurance of data definition and format consistencies within and across applications.

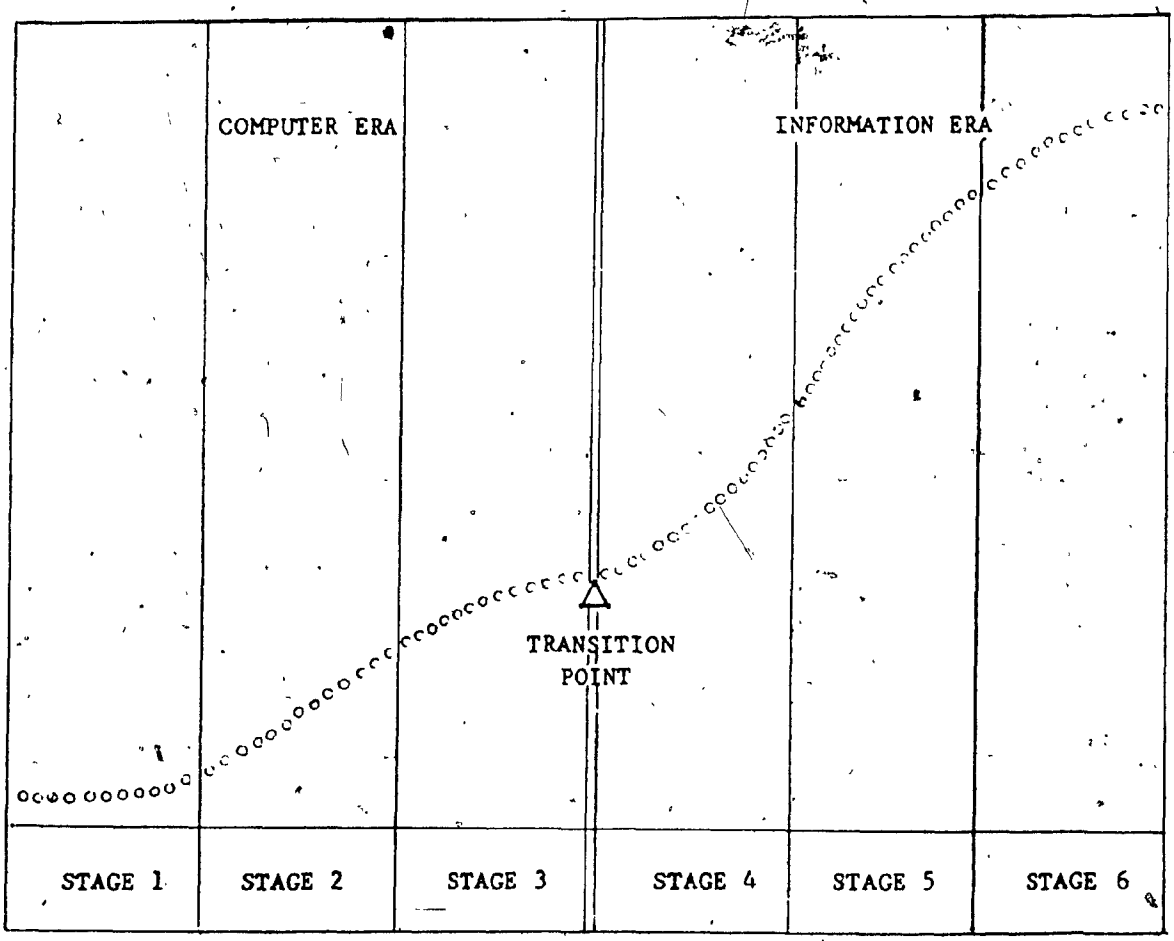
After some ten years of existence, the database approach as understood and applied today appears to be only a partial solution to the data management problem.. Reason: functionally autonomous DBMS implementations have led to the

proliferation of databases with much data redundancy, inconsistency and incohesion. The creation of an integrated corporate database in conjunction with a corporate data dictionary is not a viable solution either. The reason is that such integrated databases are extremely complex to design, build and maintain. Furthermore, lack of supporting technology makes such a dream impractical [Brown,1976; Nijssen,1978; Sundgren,1978]. A pragmatic solution to the data management problem is a set of databases functionally structured in a highly modular fashion which fit together to form, logically, an 'integrated corporate database'. The modularization stated here is only possible through an overall plan which reckons data as a corporate level shared resource and the need for its centralized control [De Blasis & Johnson,1977; Davenport,1980; Martin,1980; BSP,1980; Finkelstein,1981].

The EDP growth within business organizations, from infancy to a stage of maturity has been well summarized by R.L. Nolan [Nolan,1965]. Accrding to Nolan, this EDP growth (as depicted in Figure 1.1) comprises of six stages:

STAGE 1 - INITIATION: Marks the introduction of the firm's first computer and the introduction to automation

STAGE 2 - CONTAGION: Marks the rapid proliferation and growth of computer systems in the organization as automation



LEGEND:

Stage 1 - Initiation, Stage 2 - Contagion, Stage 3 - Control, Stage 4 - Integration, Stage 5 - Data Administration, Stage 6 - Maturity

FIGURE 1.1 SIX STAGES OF DATA PROCESSING GROWTH

spreads

STAGE 3 - CONTROL: Marks the management attempts to contain the rapidly rising costs of computing services and to bring data processing growth under control

STAGE 4 - INTEGRATION: Marks the identification of data redundancy problems and solutions (controlled systems growth and integration of diverse systems/technologies)

STAGE 5 - DATA ADMINISTRATION: Marks the conclusion of the development and implementation of completely integrated database systems, and

STAGE 6 - MATURITY: Marks the final stage of data processing maturity, i.e., the application systems plan is completed and its structure 'mirrors' the organization and the information-flows in the company.

At some point in stage 3, there is a transition from computer management to information management. Synott [Synott,1981] calls the first three stages the computer era and the last three, the information management era. Most well run DP organizations have now been through the computer era, and are now at the transition point of moving into the information management era.

1.3 DATA ADMINISTRATION - BASIC CONCEPTS

If we study the dynamics of data within an enterprise, we can identify a dataflow network inherent to the enterprise. In such a network the nodes represent business processes and arcs represent the flow of a class of data. In most organizations this data flow network is not optimal; because the network is not based on the enterprise's overall data needs but rather based on the local needs of processes or process clusters. It is also possible that the dataflow within each process cluster is far from optimal; for, local dataflow may exist based on 'tradition', 'convenience' or simply 'as a matter of chance'. This situation is intolerable if the enterprise's objectives are: to view data as a valuable resource, to promote its sharing corporation-wide, and to avail timely/accurate/reliable data to upper management for sound decision making purposes.

The foregoing discussion has identified the need for data resource control from the top, i.e., 'corporate data administration'. The advantages of 'centralized resource control' as implied by data administration are:

- o control of redundant data within an enterprise
- o reduction in cost of doing business through optimal use of data
- o untapping of hidden data which is of value to the enterprise
- o provision of adequate security/privacy to data, and

- o ensurance of timely, accurate and reliable information to top management

Data administration is a corporate endeavour and comprises of the following components:

- o a data administration infrastructure
- o a corporate information system plan, and
- o an information systems plan generation, implementation and support methodology (i.e., data administration methodology)

The ultimate goal of data administration is to gain control of corporate data and to ensure its optimal corporation-wide usage, through data-sharing.

1.3.1 DATA ADMINISTRATION INFRA-STRUCTURE

The data administration function consists of two major components:

- o Meta-data management, and
- o Data management

Figure 1.2 describes the data administration function hierarchy.

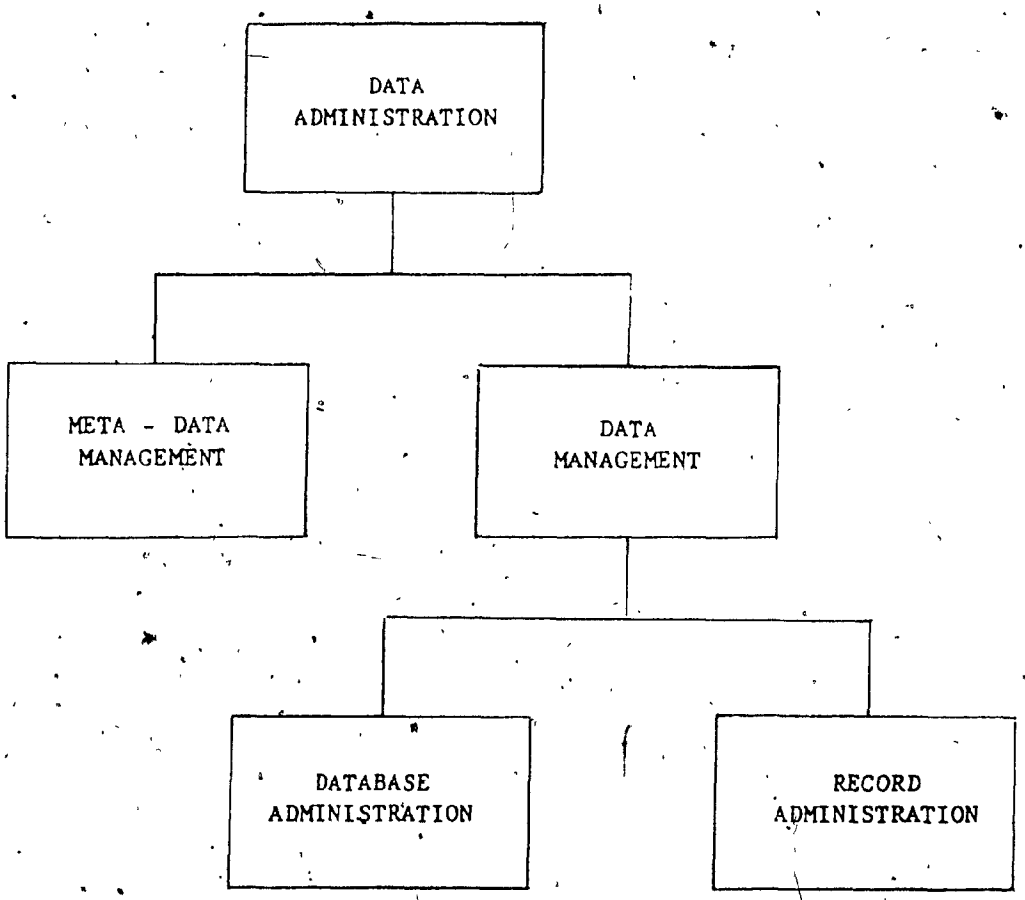


FIGURE 1.2 DATA ADMINISTRATION FUNCTION

META-DATA MANAGEMENT

Meta-data management is a centralized function which is partly business oriented and partly technically oriented [Davenport,1980]. Meta-data management concerns itself with: business function analysis, business modelling, definition of entities and relationships, conceptual data modelling, specification of data privacy types, definition of data integrity-constraint types, and definition of data valueset types, etc.

A major concern of meta-data management, is the development and maintenance of the conceptual data model for the enterprise. This model describes the entities of interest to the enterprise and the relationships between them. The overall responsibility for meta-data management lies on the 'Data Administrator' - it is convenient to reckon this position as being a staff group. The data administrator liaises with the database administrator and the records administrator (see DATA MANAGEMENT) to ensure the overall realization of the corporate information systems plan (see Section 1.3.2). The meta-data management responsibilites encompass the following functional areas [Finkleston,1980]:

- o Overall Guidelines, i.e., policies, procedures, guidelines, standards, strategic plans, etc.
- o Documentation of Models, i.e., business model, conceptual data model, information model, data

access model, data distribution model, etc.

- o Tools and Techniques, i.e., data dictionary, data languages, query languages, utilities, etc.
- o Control Procedures, i.e., manual procedures and standards, automated controls, etc.

The meta-data management activities are carried out in close co-ordination with end-users, business analysts, systems analysts, database administrator and records administrator.

DATA MANAGEMENT

Data management is responsible for the automated, as well as, the non-automated portions of the corporate data. On functional grounds, the data management function can be viewed as having two components [Davenport,1980]: database administration and records administration.

Database Administration:

Database Administration is a technically oriented rather than a business oriented function. This function component supports and complements the meta-data management function. The database administration concerns itself with: internal (storage) data model, external (user) data model, data access control, data security, backup and recovery of stored data, database software maintenance, database applications

support, database monitoring and tuning, etc.

The database administration function is headed by the database administrator (DBA) - this position could well be considered as being a staff group. Primarily, the DBA is responsible for the definition of internal data model and optimal physical storage sturctures for data to provide acceptable level of machine performance. The DBA is also responsible for the external (user) data model which is required to support user activities. The DBA liaises with the database users and is guided by the policy decisions made by the data administrator.

Records Administration:

Records administration is a business oriented function and complements the database administration function. The records administration concerns itself with the capture and storage of all non-mechanized data required by the organization, its security, control and archiving.

The records administrator function is headed by the records administrator (RA). As in the case of database administration, this position can be thought of as being a staff group. The RA liaises with the end-users and business analysts, and is guided by the policy decisions of the data administrator.

1.3.2 CORPORATE INFORMATION SYSTEMS PLAN

A Corporate Information Systems Plan is a strategic plan that views the enterprise's role and its need for data from a global perspective. The Corporate Information Systems Plan addresses undertakings that are strategic and long term in nature [BSP,1981]. The salient features of such a plan are:

- o it is consistent with the corporate goals, objectives, and long term plans
- o it identifies the functions/processes that are essential to the success of the business, and their data needs
- o it provides a 'roadmap' that identifies system development strategies and migration paths
- o it describes conceptually understandable and modularly implementable and extensible information systems
- o it identifies information systems plan implementation priorities

The corporate information system plan is like an architect's 'blueprint' for a large construction project. It is according to this blueprint, the current/future information systems are built and eventual operational systems are integrated. With such a plan the potential of data can be fully exploited organization-wide.

1.3.3 DATA ADMINISTRATION METHODOLOGY

A methodology which addresses both, the development of an Informaton Systems Plan and its implementation can be termed as a Data Administration Methodology. A data administration methodology should define the following:

- o the data administration infra-structure, roles and responsibilities
- o an overall approach for the corporate Information Systems Plan development and implementation
- o a list of data administration activities and their execution sequence
- o data administration products
- o dependencies/relationships between data administration activities and products
- o techniques and automated/manual tools that are required

Currently, there exist few integrated data administration methodologies [Leong-Hong,1978a; Quintella,1979; Martin,1981; Finkelstein,1981; Sakamoto,1982; Zachman,1982]. However, there do exist a number of ad hoc methodologies in the DA subject areas: Strategic Planning, Systems Analysis/Design, Data Modelling, and Database Design. The applicability of these methodologies to a particular business organization depends on the inherent characteristics of the organization. In

some cases an in-house developed hybrid methodology might turn out to be the most appropriate. The data administration methodology topic is further discussed in Chapter 2.

1.4 DATA ADMINISTRATION - FUTURE IMPACT

An organization's commitment to data resource management has the following implications [Finkelston,1980]:

- o formal recognition of the data administration function
- o data administration staffing
- o organization-wide education
- o data processing user cooperation
- o systems development methodology changes
- o organization-wide enforcement of policies, procedures and standards concerning the data resource
- o hardware/software expenditures
- o full management support

Data resource management is a large undertaking and will affect all parts of a business organization. The organization-wide data sharing might lead to totally a new way of doing business: new organization structures might become necessary, existing procedures might not work, creation of business procedures might be required to take advantage of new opportunities, derivation of integrated

procedures that cross organizational boundaries might be necessary, etc. Enterprise analysis may also highlight opportunities for corporate restructuring to benefit from the new insights provided [Martin,1981; Synott,1981].

Other factors that will have a major impact on data administration are the developments in the areas of hardware, software and communication. The availability of high bandwidth communcication facilities, vast amount of mass storage, data distribution capabilities and sophisticated intelligence will soon surmount many of the technological constraints that existed in the past. The data administration endeavour will bring together the management, end-user and DP personnel in a new partnership that will exploit the unique skills and experiences of each [Finkelstein,1981]. The data models created will no more be DP deliverables but rather products created by the users and management who have a far greater understanding of the organization - these models will provide precise communication of requirements to DP for their physical manifestation and support. Data administration, therefore, will play a key role in the implementation and support of sophisticated computer based information systems in the future. The corporate data administration function will be an absolute necessity as data processing becomes more mature throughout the 1980's [Stanfield,1980; Parker,1982]. The Data administration topic is furter discussed in Chapter 2.

CHAPTER 2

DATA ADMINISTRATION - CONCEPTS, METHODOLOGIES AND TOOLS

2.1 DATA ADMINISTRATION CONCEPTS

A familiarity with the data administration concepts can be best obtained through an understanding of the aggregate concept - 'enterprise modelling'. In this section we will first examine 'real world modelling' and the 'entity-relationship data modelling approach'. We will then use this knowledge to understand 'enterprise modelling', and its relevance to data administration.

2.1.1 REAL WORLD MODELLING

A perception of the real world can be regarded as a series of observed phenomena [Tsichritzis,1982]. The description of the observed phenomena constitutes data, which enhances our knowledge of the perceived world. A means to augment this knowledge of the perceived world is 'interpretation'. But to do this we need an abstraction tool to put together the details that we have gathered. We call such a tool a 'model'. A model should closely fit the reality, and should be flexible enough to absorb minor perturbations in the real world. A roadmap is an example of such a model - it captures a part of reality (i.e., roads,

highways, rivers, bridges, etc.) and yet is stable with respect to minor incidents in the real world. To cite an example from queuing theory,

$$Q = \frac{\lambda^2}{\mu(\mu - \lambda)}$$

is a mathematical model that helps us to interpret a part of reality comprising of average queue length (Q), mean arrival rate (λ) and mean service rate (μ). Within the realm of this model, the observed phenomena pertaining to queue length, mean arrival rate and mean service rate fit together. Further, the model provides us a 'rationale' on the basis of which we can interpret a subset of the real world.

Models are extensively used in many disciplines. We have physical models to interpret physical properties of the real world, mathematical models to abstract the real world using symbols and logic, economic models to simulate/predict economic trends, etc. Though data modelling is relatively new, according to recent surveys [Kerschberg,1976; Senko,1977] there exist some thirty data models and more are being proposed. The proliferation of data models cannot be controlled by searching for the best data model. Just as it was eventually recognized in programming languages that there is no 'best' programming language, so it is now generally conceded that there is no one 'best' data model

[Tsichritzis,1982]. What we have to really look for is a common denominator on the basis of which we can derive and interpret any given model. The common denominator for all existing data models appears to reside in the conceptual approach to data model structures, i.e., in an approach which is focussed on semantic aspects, rather than efficiency and/or user convenience aspects [Falkenberg,1978]. Since different applications have different data modelling needs it is essential to allow co-existence of data models [Nijssen,1976; Kerschberg,1976; Falkenberg,1978; Tsichritzis,1982]. Given an application, the appropriateness of any particular model largely depends on the problem in hand. In other words, a model which is suitable for enterprise analysis problem may not be a candidate for an artifical intelligence problem.

In the area of enterprise modelling, one of the better known and widely used data model is the entity-relationship or E-R data model [Chen,1976; Chen,1979]. The E-R modelling approach is described in the next section.

2.1.2 ENTITY-RELATIONSHIP DATA MODELLING APPROACH

The Entity-Relationship (E-R) Data Model [Chen,1976; Chen,1977a; Ullman,1980; Tsichritzis,1981] of Chen is an extension of data structure diagrams proposed by Bachman [Bachman,1975]. The E-R model is easy to conceptualize and is intutively appealing. In the E-R model the real world is

viewed in terms of 'entities' and 'relationships' between entities. Real world objects are not dealt with directly, rather user level 'name structures' are operated upon [Senko,1978]. This user level is similar to 'discourse system' [Stamper,1973] and 'infological level' [Langefors,1980].

E-R MODEL DEFINITIONS

Enterprise:

An enterprise is a subset of the real world that is of interest to a group of people.

Example : a company, say, Bell Telephones.

Entity:

An entity is an 'object', 'thing' or 'concept', real or abstract, that is of interest to an enterprise.

Example: a specific employee, say, John Doe.

Entity Set (Entity Type):

An entity set is an abstraction representing a class of entities - the class is determined by the attributes of the entities in question.

Example: 'employee' is an entity type which stands for a group of people employed by an organization.

Relationship:

A relationship is an association between two or more entities, which is of interest to the enterprise.

Example: 'employee-project' is a relationship between two entity types, 'employee' and 'project'.

Relationship Set (Relationship Type):

A relationship type is an abstraction representing a class of relationships that connect one or more entity types. A relationship that connects n entity types is called a n-ary relationship.

Example: The relationship 'employee-project' connecting entity types 'project' and 'employee' is a binary relationship.

Note:

A relationship set, Ri, is a set of mathematical relations [Birkhoff,1970] among n entities, of the form:

$$Ri = \{(e1,e2,e3,\ldots,en) \mid e1 \in E1, e2 \in E2, \ldots, en \in En\}$$

Here, each tuple (e1,e2,...,en) represents a relationship and E1,E2,...,En are entity sets.

Weak Entity Set (weak entity type):

An entity set can have an independent existence or its existence can be dependent upon the existence of another

entity set. In the latter case, the dependent entity set is called a 'weak entity set'. A weak entity set is associated with its parent entity set by an existence dependent relationship.

Attribute:

A distinguishing property of an entity or relationship is called an attribute of that entity or relationship.

Example: 'name', employee#', 'date-of-birth', 'sex' are typical attributes of the entity 'employee'.

Valueset (domain):

A valueset is a collection of values that an attribute can assume.

Example: Attribute 'colour' of entity 'car' can have a valueset {blue,green,yellow,orange,red}.

Note:

An attribute can be formally defined as a function, which maps from an entity set (or a relationship set) into a valueset(domain) or a cartesian product of valuesets:

fi : Ei ----> Vi or

f : Ei ----> Vi1 x Vi2 x ... x Vin ; $1 \leq i \leq N$

Here, Ei represent entity sets and Vi or Vij represent valusets.

E-R MODELLING PROCESS

The E-R model of an enterprise can be derived through a method of investigation of the real world. This process of modelling is characterized by three steps [Moulin,1981]:

- o Perception of the real world - identifying entities and relationships
- o Induction - discovering properties of entities and relationships
- o Abstraction - classifying entities and relationships and building a structure that captures the semantics of the perceived world

At the perception step we identify entities and relationships between these entities unique to an enterprise's universe of discourse. The objects and concepts of the real world that we try to model may not readily fit into one of the three categories: entity, attribute and relationship [Kent,1977; Shave,1981].

Example:

The concept 'marriage' could mean:

- o an entity type with attributes such as date, place, name of bride, name of bridegroom
- o an attribute type associated with an entity, say, person, or
- o a relationship between two entity types: 'man' and 'woman'

All theoretically possible relationships may not be

meaningful or of interest to an enterprise. The relationships between entities can take several forms - multiple relationship types can exist between the same two or more entity types, a relationship can exist between the same entity type (i.e., involutary relationship), a relationship could be existence-dependent on the participating entities, etc. Also, a relationship can have a 'degree of relationship' associated with it, such as (1,1), (1,n), (n,1), or (n,n).

At the induction step, we invoke a mental process to identify the properties of entities and relationships - this provides us a clearer understanding of the 'object' of the perceived world.

At the abstraction step, the entities and relationships identified earlier are re-examined. This may lead to identification of new entities/relationships, splitting of an entity type into more simpler entity types, decomposition of a relationship into less n-ary other relationships, etc. This process ensures that the E-R model, which is built next, holds the most fundamental objects of the enterprise that is being modelled.

ENTITY-RELATIONSHIP DIAGRAM

A graphic technique is used to represent entities and relationships in the E-R model. The graphic representation

is called an entity-relationship diagram (ERD). The convention used for representing entities and relationships is actually an extension of data structure diagrams [Bachman,1969]. Figure 2.1 describes the symbols used to draw ERD's.

As shown in Figure 2.1, an entity is represented by a rectangle and a relationship by a diamond. The participation of an entity type in a relation type is indicated by an arc connecting the respective symbols. The degree of the relationship is denoted by a number beside each arc. A valueset is represented by a circle; an attribute is represented by an arc connecting the entity (or relationship) to the appropriate valueset.

An ERD can be viewed as consisting of two domains [Chen,1977]:

- o An upper conceptual domain consisting of entities and relationships, and
- o A lower conceptual domain consisting of attributes and valuesets

The upper and lower conceptual domains concept is described in Figure 2.5.

Examples of E-R Diagrams:

Example 1: Figure 2.2 shows an ERD of relation 'employee-project' between entities 'employee' and

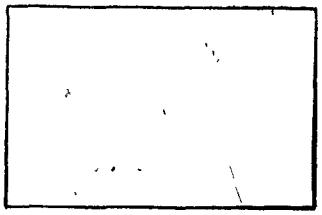
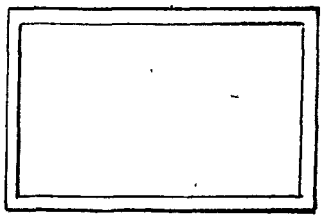
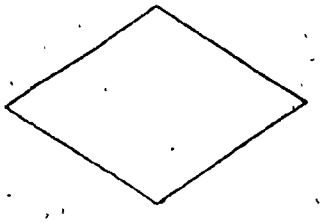
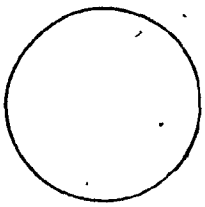
SYMBOLS:	MEANING:
	ENTITY
	WEAK ENTITY
	RELATIONSHIP
	ATTRIBUTE
	ATTRIBUTE VALUESET (DOMAIN)

FIGURE 2.1 ENTITY-RELATIONSHIP DIAGRAM (ERD) SYMBOLS

'project'.

Example 2: Figure 2.3 shows an ERD of an involutary relationship 'supervises' between entity 'employee' and itself.

Example 3: Figure 2.4 shows an ERD of a dependent relationship 'employee-child' between entities 'employee' and 'child'.

Example 4: Figure 2.5 shows an ERD depicting the attributes, valuesets, upper and lower conceptual domains associated with the relationship 'employee-project' between entities 'employee' and 'project'.

E-R MODEL EXTENSIONS

Due to its wide popularity, the E-R model has been a target for several extensions/modifications in recent years [Benci,1976; Bachman & Daya,1977; Ledgard,1977; Smith,1977; Chen,1979a; Chiang,1979; Davenport,1979a; Lusk,1979; Hawryszkiewycz,1980]. Some of the significant issues raised are:

- o Inclusion of spatial and temporal specifications to entities, attributes and relationships [Benci,1976; Bubenko,1980]
- o The recognition of the object role concept and

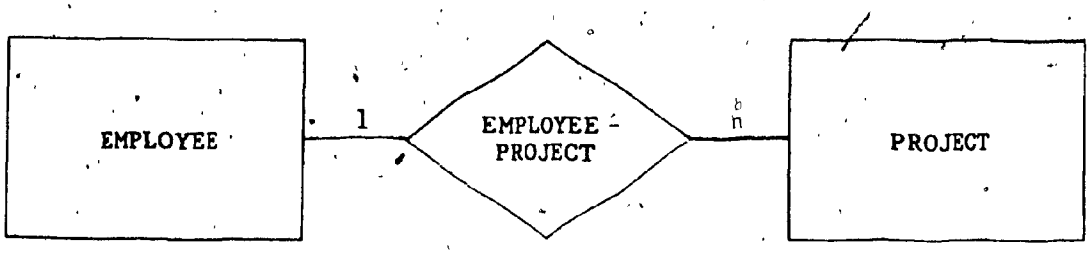


FIGURE 2.2 ERD: EMPLOYEE-PROJECT RELATIONSHIP

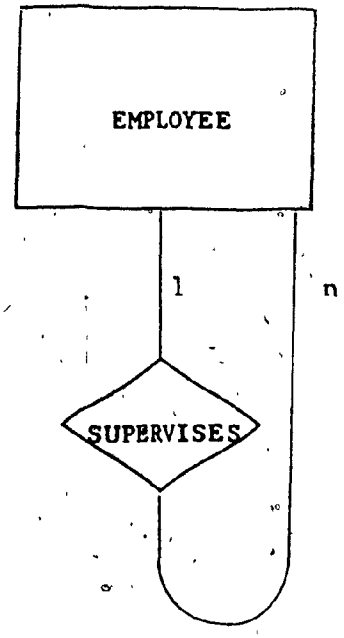
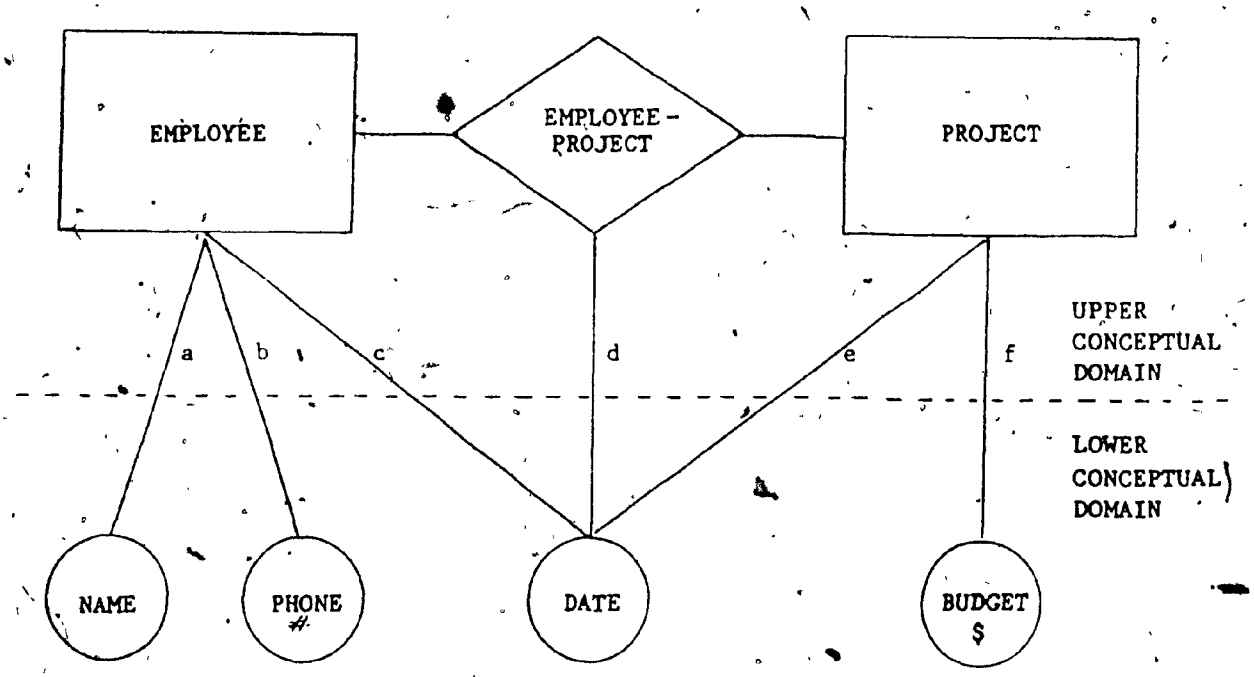


FIGURE 2.3 ERD: INVOLUTARY RELATIONSHIP

FIGURE 2.4 ERD: DEPENDENT RELATIONSHIP



LEGEND:

Arcs a, b, c, d, e and f represent attributes 'name', 'telephone#', 'date-of-birth', 'project-assignment-date', 'project-deadline' and 'project-budget', respectively.

FIGURE 2.5 ENTITY-RELATIONSHIP CONCEPTUAL DOMAINS

modelling of the real world using just two concepts - 'object' and 'role' [Falkenberg,1978]

- o The recognition of 'entity' and 'entity role' as distinct concepts in modelling the real world [Bachman & Daya,1976]
- o Restriction of attributes only to entities, i.e., no attributes for relationships [Davenport,1978]
- o Use of abstraction which permits relationships between named objects to be thought of as a higher level named object [Smith,1977]
- o Inclusion of integrity constraints at the attribute valueset level [Machgeels, 1976]
- o Recognition of 'categories' concept and that entities belonging to many categories [Kent, 1976]

It is to be noted that a model, being an artificial formulation, does not necessarily fit exactly the target real world subset (enterprise). Sometimes it becomes necessary to perceive the real world a little differently to achieve a perfect fit with the model.

2.1.3 ENTERPRISE MODELLING

An enterprise is a finite subset of the real world that is of interest to a group of people. An enterprise could be an entire company, a functional area within a company, a process and so forth [Hubbard,1981]. It is modelled in terms of objects, concepts and

interrelationships existing within. The purpose of enterprise modelling is to capture the enterprise semantics in a formulation that is easily representable by computers. Closer a model is to reality, simpler it is to draw meaningful interpretation out of our target world. Enterprise modelling is a problem of considerable complexity. It is difficult to capture the total semantics of an enterprise in a single model. Therefore, the current approach is to use a collection of models. These models include business models, data models (i.e., conceptual data model, internal data model, external data model), data distribution models, data access models, information models, etc. (note: the term 'model' and 'schema' have been used interchangeably in the following discussion).

BUSINESS MODEL

A business model is a formulation which describes the business processes and their interrelationships. It is a high-level representation at the business function level showing the processes involved in the conduct of an enterprise's activities. Figure 2.6 describes a business model for the 'order-process' function.

DATA MODELS

A Data model depicts the inherent nature of data and inter-relationships among data entities. The current

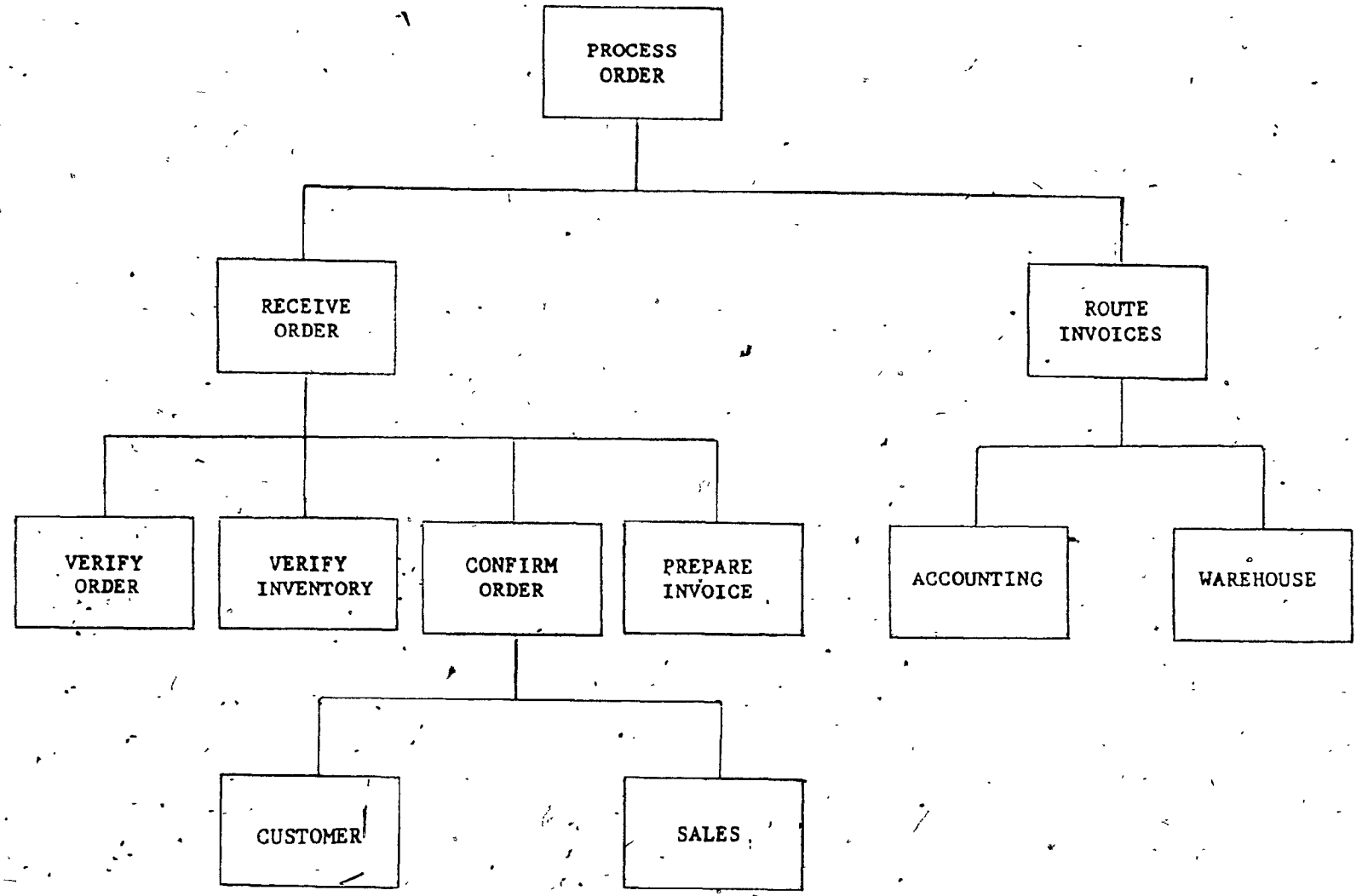


FIGURE 2.6 BUSINESS MODEL : ORDER PROCESSING

approaches to data modelling have been considerably influenced by the 'three-schema architecture' approach proposed by ANSI/SPARC and CODASYL/DBTG [ANSI,1975; Gerritsen,1975; Taylor,1976; Senko,1976; CODASYL,1978; Manola,1978; O'Connell,1978; Clemons,1979]. ANSI/SPARC - American National Standard Institute/Standards Planning and Requirement Committee proposed a three-schema architecture for the database systems, in 1975. The proposals by CODASYL/DBTG - COnference on DAta SYstems Languages/Data Base Task Group are similar to those of ANSI/SPARC [CODASYL,1971 & 1978]. The three-schema architecture consists of:

- o a Conceptual Schema (or enterprise schema) - conceptual view of an enterprise's data
- o an External Schema (or user schema) - description of end-user's views of data
- o an Internal Schema (or storage schema) - description of physical data organization in storage devices

Figure 2.7 describes the relationships/interfaces between the three schemas described above. The conceptual, external and internal models are further explained in the following sub-sections.

Conceptual Data Model:

A conceptual data model (enterprise schema) is a description

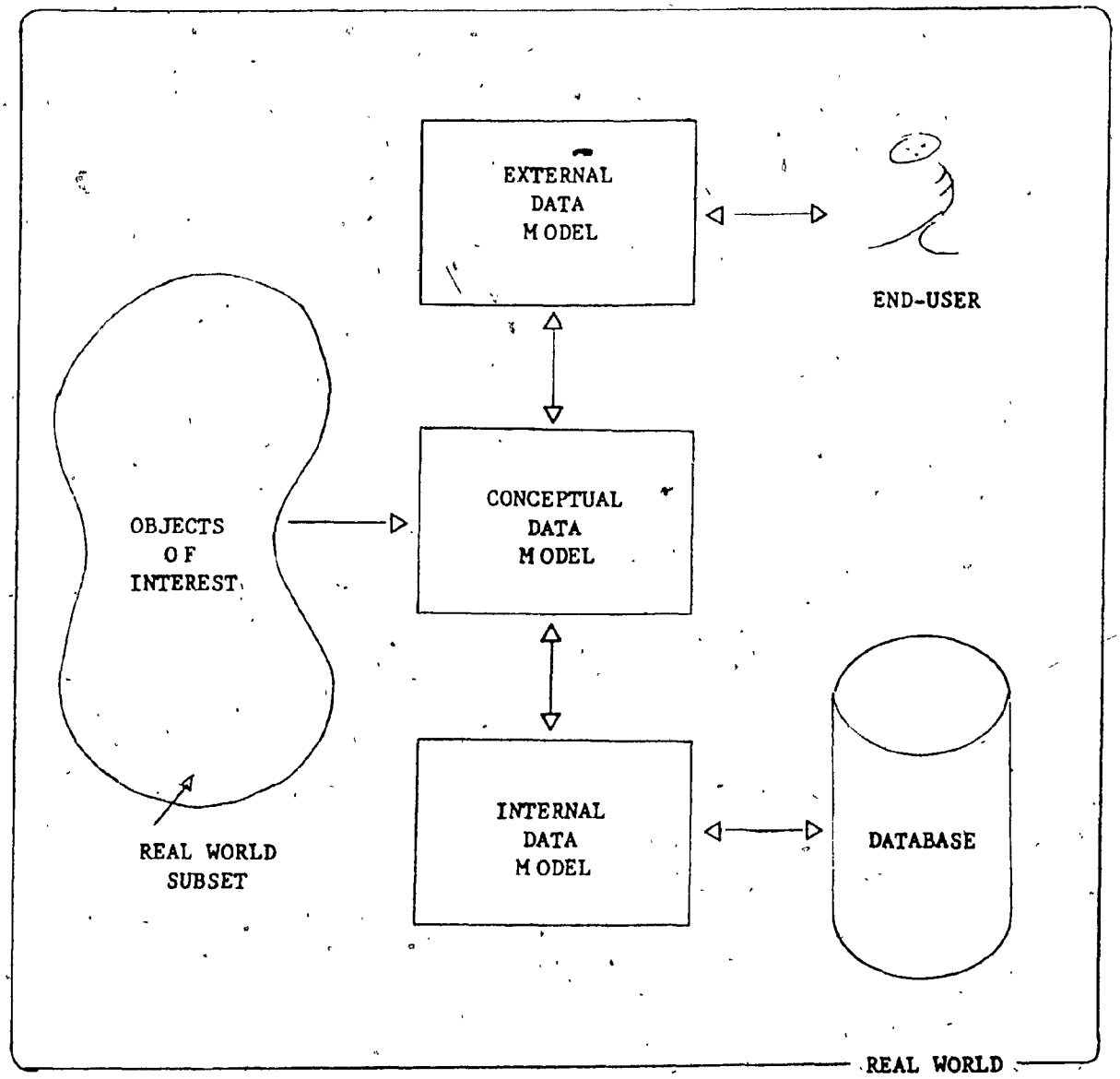


FIGURE 2.7 (ANSI) THREE-SCHEMA ARCHITECTURE

of the semantics of an 'enterprise', employing concepts equivalent to 'entities', 'relationships' and 'attributes'. The ANSI/SPARC definition of the conceptual schema states: It represents the enterprise's view of the structures it is attempting to model in the database [ANSI,1975].

The conceptual data model, because it deals with broad concepts of an enterprise, remains largely invariant with respect to its semantics. Of course, the conceptual data model grows - as new business functions are added, the local conceptual data models are integrated with the global conceptual data model. The latter process is depicted in figure 2.8.

The conceptual data model is devoid of any influences from the current DBMS implementations. We generate the external and internal data models from the conceptual model using appropriate mapping mechanisms.

External Data Model:

External data model is a comprehensive problem oriented data structure that can be chosen by the end users. It is generated from the conceptual data model through a mapping mechanism. The external data model can be expressed in terms of different data structure types:

- o Hierarchical - views real world objects in the form

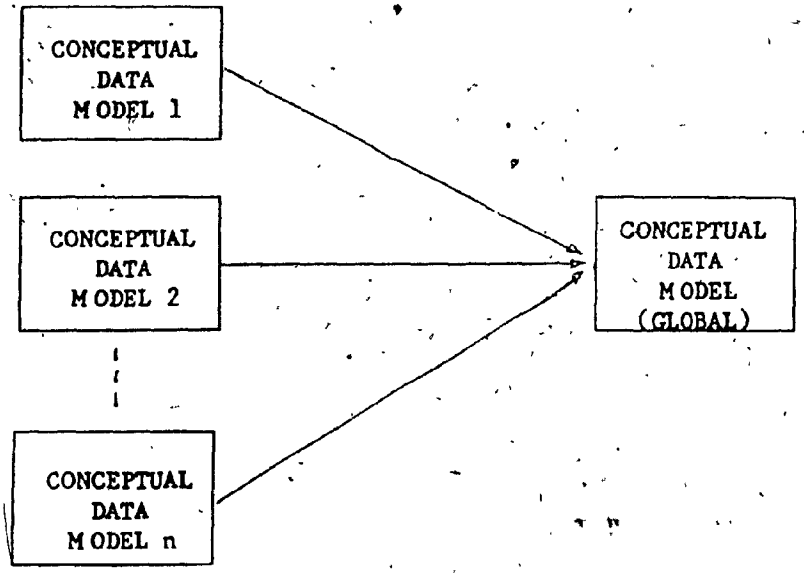


FIGURE 2.8 INTEGRATION OF DATA MODELS

of a hierarchy

- Network - views real world objects in the form of a network
- Relational - views real world objects in the form of a set of tables.

Figure 2.9 describes the context of hierarchical, network and relational data models within the three-schema architecture.

Internal Data Model:

The internal data model defines the physical organization of data within the database. The internal data model is generated from the conceptual data model via a mapping mechanism. In reality, the global internal data model is a composite of several functional internal data models generated by the mapping process.

INFORMATION MODEL

An information model is a representation of the end-user perspective of information needs. The information model provides a single cohesive, consolidated representation of end-user information requirements. The model is useful in determining the information systems necessary to satisfy the end-user needs and the subject databases required to support these information systems.

Note:

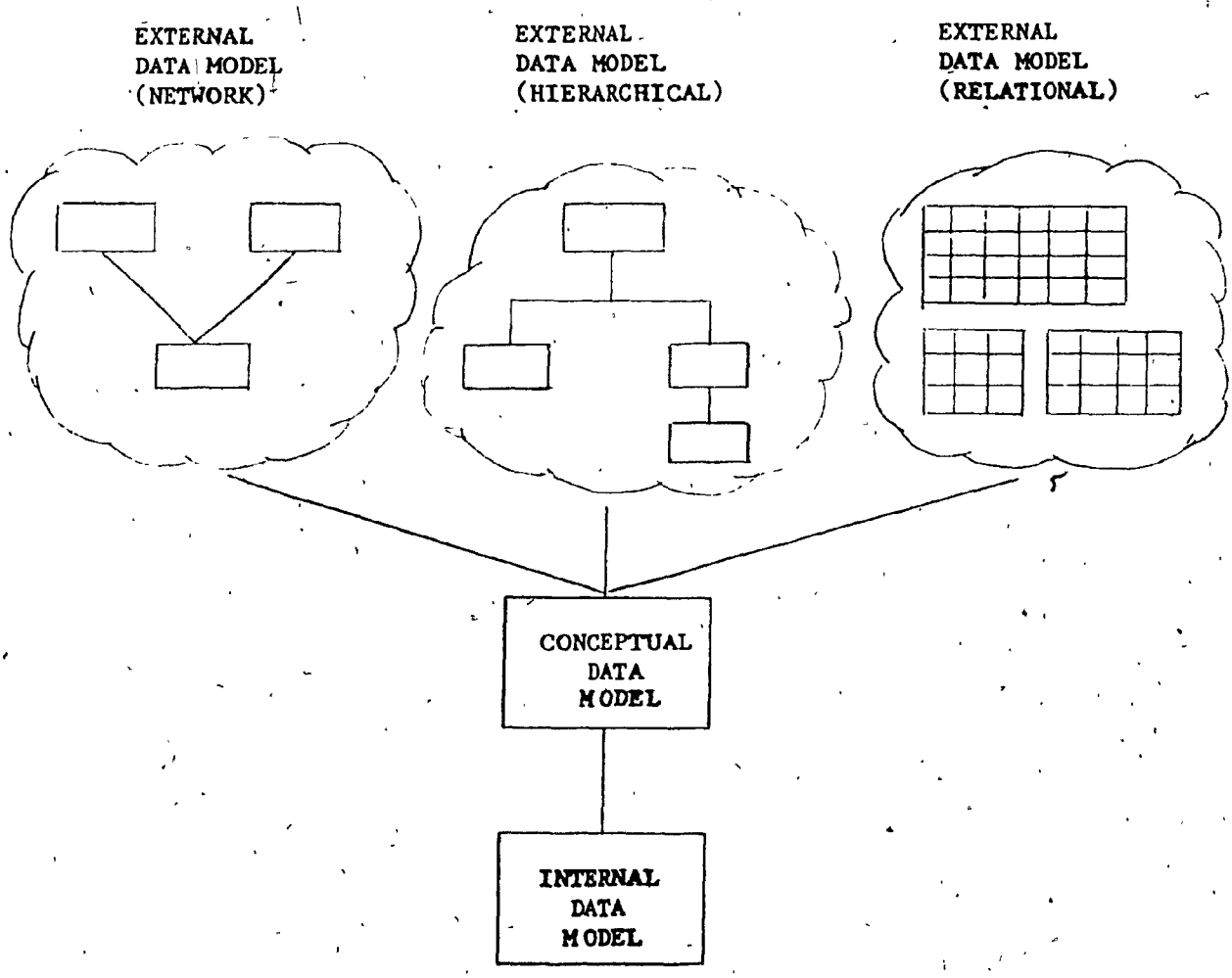


FIGURE 2.9 EXTERNAL DATA MODEL

Subject Databases are databases built on organizational subjects. The organizational subjects are identified in the strategic data planning process through entity affinity analysis.

DATA ACCESS MODEL

A data access model describes the framework for data access. It depicts the subject databases and the type of access by business functions. The model is useful in determining the users of any given database and ownership responsibily for the data.

DATA DISTRIBUTION MODEL

A data distribution model shows the usages of each subject database by user staff groups. The model is useful in determining the required physical distribution of databases.

2.2 DATA ADMINISTRATION METHODOLOGY

The management structure of a typical business organization, comprises of three levels: general, functional, and operational [Anthony,1965]. The scope of each level is such that 'operational' management encompasses management of the day-to-day acivities, 'functional' management encomapasses management of a business unit, and

'general' management encompasses the management of the entire enterprise. The three management levels described above are characterized by the required planning and control levels.

The Planning and Control Levels in Management are:

- o Startegic Planning - process of deciding on objectives of the organization, on changes to these objectives and on policies that are to govern the acquisition, use and disposition of these resources
- o Management Control - Process by which managers assure that resources are obtained and used effectively in the accomplishment of the organization's objectives
- o Operational Control - Process by which the managers assure that specific tasks are carried out effectively and efficiently

In Figure 2.10, a pyramid is used to illustrate the groupins of systems that support the planning and control levels. The operational management systems are standalone systems. They are characterized by multiple acquistion of data and serial distribution of that data (e.g., an order entry system). On the other hand, management control systems encompass both functional and operational levels of management. These systems by their very nature are data-managed, i.e., characterized by consolidated acquistion

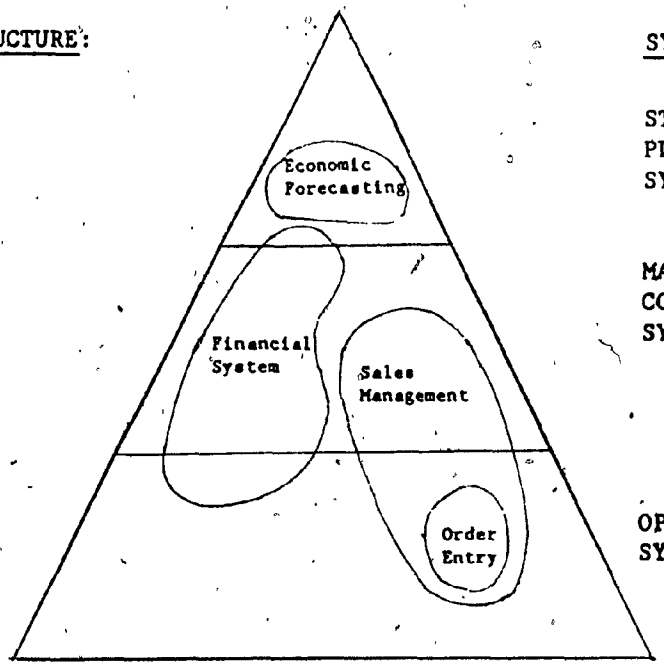


FIGURE 2.10 BUSINESS MANAGEMENT LEVELS AND MANAGEMENT LEVEL SUPPORT SYSTEMS

of data, controlled data stores, and parallel data distribution (e.g., a financial system). Initially, most data processing systems were concentrated at the operational level, e.g., order entry, cheque processing, inventory systems, etc. This is because the labour-intensive operations of an enterprise provided the best financial justifications for mechanization. Over time, the EDP sphere of influence has widened - in recent years there has been a steady growth of data-managed systems directed towards tactical management of corporate resources. The data-managed systems call for strategic planning at the general management level. The strategic planning is necessary to address such issues as data acquisition, data storage, data distribution - in other words data administration.

With current focus on data administration the need for a data administration methodology has been well realized by most well run business organizations. The basic components of a Data Administration (DA) Methodology are:

- o an information system plan (ISP) generation methodology
- o an ISP implementation methodology

The overall DA methodology should therefore, not only assist in developing a corporate information systems plan, but also provide guidance for its implementation. The known methodologies that meet the above criterion are only a few

[Martin,1981; Finkelstein,1981]. On the other hand, there exist numerous ad hoc methodologies in the DA subject areas:

- o STRATEGIC PLANNING [BSP,1981; Flavin,1981; Martin, 1981; BICS,1982]
- o SYSTEMS ANALYSIS & DESIGN [Head,1971; Warnier, 1974; Jackson,1975; Bech,1976; Gane & Sarson,1977; De Marco,1978; Nunamaker,1978; Yourdon,1978; Chand,1980; Orr,1981]
- o DATA MODELLING [Chen,1976; Adiba,1976; Bracii, 1976; Smith,1977; Bubenko,1977a,b; Batini,1979; Davenport,1979b; Oren,1979]
- o DATA BASE DESIGN [Mitoma,1975; Hubbard,1975; Khan, 1976; Finneran,1977; Raver,1977; Senko,1977; Chen,1977b; Vincent,1977; Yeh & Roussopoulos,1978; Steel,1978; Sundgren,1978; Tardieu,1979; Davenport,1979a; Tozer,1979; Teory,1980; Hawryszkiewycz,1980; Martin,1980; Atre,1980; Holland,1982]

Consequently, business organizations faced with the DA methodology selection problem have the following options:

SELECTION OF AN EXTERNAL METHODOLOGY

Information Engineering [Finkelstein,1981] methodology of Information Methods Incorporated is an example of this option. This approach has advantages because the

constituent methodologies are sufficiently tested, and necessary training and application support is provided by the vendors.

DEVELOPMENT OF AN 'IN-HOUSE' METHODOLOGY

This involves adapting a known methodology or methodologies to form an hybrid data administration methodology that will suit the local needs. Advantages of such an approach are: it can be tailored to address inherent characteristics of the organization, and being an 'in-house' methodology, it could offer greater flexibility in its application.

2.2.1 ANATOMY OF A DATA ADMINISTRATION METHODOLOGY

As pointed out earlier, a data administration methodology has two components:

- o Information Systems Plan generation methodology (i.e., Strategic Planning Methodology), and
- o Information Systems Plan implementation methodology (i.e., Systems and Database Development Methodologies)

2.2.1.1 STRATEGIC PLANNING METHODOLOGY

The Business Systems Planning (BSP) is an example of a strategic planning methodology. BSP is a structured

approach to assist a business organization in establishing an information systems plan. The information systems plan is created using 'business processes' and 'data classes' that would satisfy the short and long-term information needs [IBM,1981]. The basic concept of BSP is top-down information system planning and bottom-up design/implementation. A management study team conducts interviews, gathers and analyses the gathered information, and prepares a written report to management. The report is used for both strategic planning and tactical planning. The BSP methodology is modular, it can be divided into several functional blocks - see Figure 2.11.

Prior to starting a BSP study, some organizational activities need to be completed. The activities include: gaining commitment from upper management, forming the team and selecting the team leader, defining the team structure and responsibilities, and planning physical facilities.

Business Objectives Definition:

This step is intended to ensure agreement among all executive levels as to where the business is going, so that information system plan can be tailored in accordance with corporate directions. The following areas are reviewed: corporate goals and objectives, state of the mechanized and non-mechanized parts of business, organizational structure, products and services, sociological/Ecological/economic

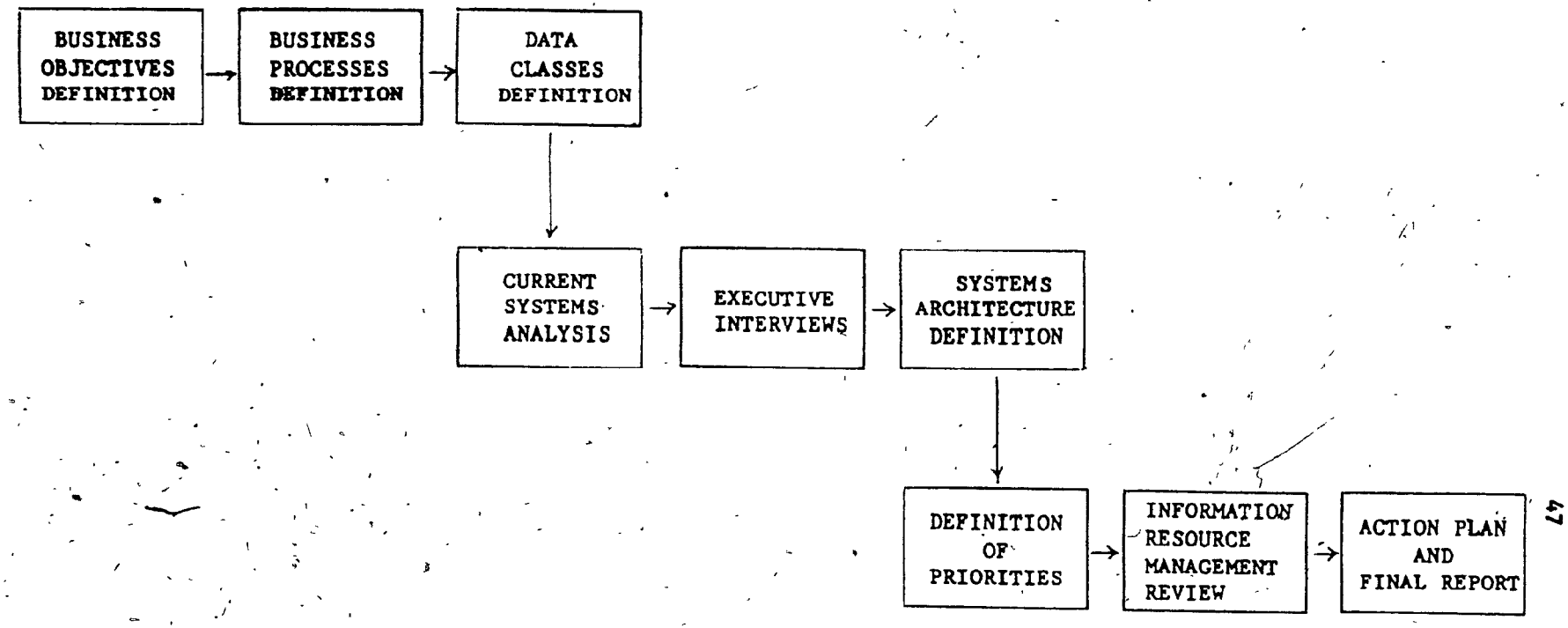


FIGURE 2.11 FLOW OF THE BSP STUDY

factors, future demands, and corporate role.

Business Processes Definition:

This step accomplishes identification of processes essential to the running and success of the business organization. The information on processes gathered in this step is used to plan/design: executive interviews, information system architecture, business problem analysis, data class identification, and various follow-up activities.

Data Classes Definition:

The objective of this activity is to group data into logically related categories. Each category supports one or more business processes. This step results in a definition of all data that is to be managed as a resource across business units.

Current Systems' Analysis:

The purpose of this activity is to study the inter-relationships or dependencies between: organization structures, business processes, existing automated applications, and data classes. Two key matrices are generated in this step using the gathered data. These matrices are: process vs organization matrix and process vs dataclass matrix.

Executive Interviews:

This step permits team members to select key corporate executives to review the planning team's accomplishment. The step provides a venue to detect inaccuracies, gain ideas, ensure executive rapport and involvement, and get credibility to the work completed thus far.

Information Architecture Definition:

In this step, the general information systems architecture (information systems plan) is defined. The information systems architecture provides a blueprint for the future information systems and their accompanying data. An enterprise level architecture is required for several reasons [Zachman,1982]:

- o to provide a scenario for identification of enterprise-wide opportunities for exploiting information technologies
- o to serve as a context within which to make the various trade-off decisions between the long-term and short-term options: current technology can support enterprise-wide, top-down system design but the practicality of resource limitations, project manageability, and the requirement to produce short-term results makes a bottom-up piece by piece approach to implementations mandatory
- o to constrain the systems design/implementation activity such that required relationships and

dependencies are identified and protected at all times

Definition of Priorities:

A total information architecture cannot be developed and implemented in a single project. The objective of this step is to assign priorities for the implementation of sub-systems and databases based on certain established criteria.

Information Resource Management Review:

The purpose of the information resource management (IRM) review is to identify the following: changes to the information systems functions that can enhance success in the follow-up projects, changes that are required to manage and implement high priority information architecture projects, and major activities that will become projects in the follow-on to the BSP study. The output from this review becomes the basis for the development of an IRM action plan.

Action Plan and Final Report:

The purpose of the IRM action plan is to assist management in its decisions regarding the recommended follow-on projects - the plan identifies specific resources, schedules and interactions between projects. The objective of the final report and executive presentation is to gain management commitment and involvement for implementing the

BSP recommendations.

BSP - REMARKS

BSP is a proven methodology and experience has shown that it can be applied to organizations, both in the public and private sectors. Recall that, the information systems architecture defined in BSP is at the data classes level. It has been observed that 'data classes' is too gross a categorization. Consequently, it has been proposed to take a finely-grained approach [Martin,1981]. This approach consists of: indentification and analysis of corporate entities, clustering of entities, identification of entity groups and entity super groups, and selection of entity super groups as potential candidates for subject data base implementations.

The non-availability of formal methodologies for implementation/support of information systems plan produced by BSP is considered as a drawback [Finkelstein,1981]. Also, BSP's focus on procedures rather than on organization, and its study of business processes without much regard to emerging technologies has come under criticism.

BSP ALTERNATIVES

The strategic planning methodologies of Information Engineering [Finkelstein,1981] are broader in scope compared

to BSP. They go beyond BSP by actively involving top management in establishing the strategic directions for the future by concentrating more on the organization and the opportunities offered by the new technologies. A striking feature of Information Engineering methodologies is the development of a corporate data model based on established strategic directions/objectives and the derivation of new procedures from it (which could lead to reorganization/restructuring of business units), to achieve the fullest benefits.

An off-shoot of BSP is a new methodology called BICS - Business Information Control Study [Zachman,1982]. BICS is based on BSP and a theory called BIAIT - Business Information Analysis and Integration Technique. The BIAIT theory proposes that the complete information handling characteristics of a business organization can be defined given an understanding of seven binary variables pertaining to the business. BICS extracts the enterprise structure from a superset of predefined data categories and relationships contained in a generalized model. The categories and relationships that are pertinent to the specific business under study are assembled to generate the enterprise schema.

Because BICS uses pre-defined components, it can be quickly generated, is reproducible, and is less dependent on the skills and understanding of the analysts. The greatest

strength of BICS is its future potential - with further research, the pre-defined enterprise structure could be based on a sound theoretical foundation. The BICS generalized model could be expanded to contain data classes which include items such as business processes, objectives, measurements, reports, forms, etc. With such a generalized model, it will be possible to produce pre-defined data designs and pre-defined functional code. Such advances will open the door for sophisticated strategic study methodologies which could automatically generate information systems from a very few variables describing the business [Zachman,1982; Parker,1982].

2.2.1.2 SYSTEMS AND DATABASE DEVELOPMENT METHODOLOGIES

The outcome of Strategic Planning is an Information Systems Plan, i.e., a comprehensive blueprint for building information systems. The planned information systems are implemented in a modular, building block fashion over time, while maintaining consistency with the overall plan, organization's business priorities, budgetary constraints, and other short-term considerations. Presently, the systems and database development methodologies are well evolved [Howden,1982].

The systems development methodologies are of two types: process centered and data centered. The process centered approach uses structured analysis to identify

processes, dataflows and datastores. The dataflow and datastore information is then used to develop program structure using structured design and programming [Gane & Sarson,1977; Yourdon,1978; DeMarco,1978; Weinberg,1978]. The data centered approach concentrates first on identifying the data structure from which a program structure is developed [Jackson,1975; Warnier,1978; Orr,1981; Finkelstein,1981]. Of late, the data centered approach has come into focus, for it has now been well recognized that data is more stable than processes. Business processes, because they are based on transient elements of business, are generally much more dynamic.

The database development methodologies follow basically two different paths: one approach first generates a global schema and then derives local views in a top down manner [Smith,1977]. The other approach first models local views and then integrates them to form a global view or global schema in a bottom-up manner [Martin,1980]. There exist several database design methodologies that fit somewhere between the two extreme approaches described above - these include methodologies proposed by Bubenko [Bubenko,1976], Khan [Khan,1976], Chen [Chen,1977], Raver & Hubbard [Raver & Hubbard,1977], Tsichritzis & Lochovsky, [Tsichritzis & Lochovsky,1978], Davenport [Davenport,1978], Teory & Fry [Teory & Fry,1978], Navthe [Navthe,1978], Housel [Housel,1979], Atre [Atre,1980], Ullman [Ullman,1980],

Hawryszkiewycz [Hawryszkiewycz,1981], Finkelstein [Finkelstein,1981], Holland [Holland,1982]. The overall database development methodology embodies the following steps: gathering of userviews of data, analysis and synthesis of userviews, definition of entities, relationships and attributes, normalization of entity/relationship records, development of a conceptual data model, analysis of database local views, generation of external data model, generation of internal data model.

2.2.1.3 COHESION OF DA METHODOLOGY COMPONENTS

One of the stumbling blocks in the data administration methodology implementation is the ensurance of cohesion between constituent methodologies, viz. strategic planning methodology, database design methodology, and systems development methodology. The cohesion problem is attributable to several factors but the main one is a missing 'big picture', i.e., lack of awareness among component methodology users as to how the individual methodologies fit together. Figure 2.12 provides a high level view of the interrelationships between the data administration methodology components. As depicted, the component methodologies work hand in hand toward top-down planning and bottom-up development of databases and application systems. The overall DA methodology is geared to the development of databases and application systems that

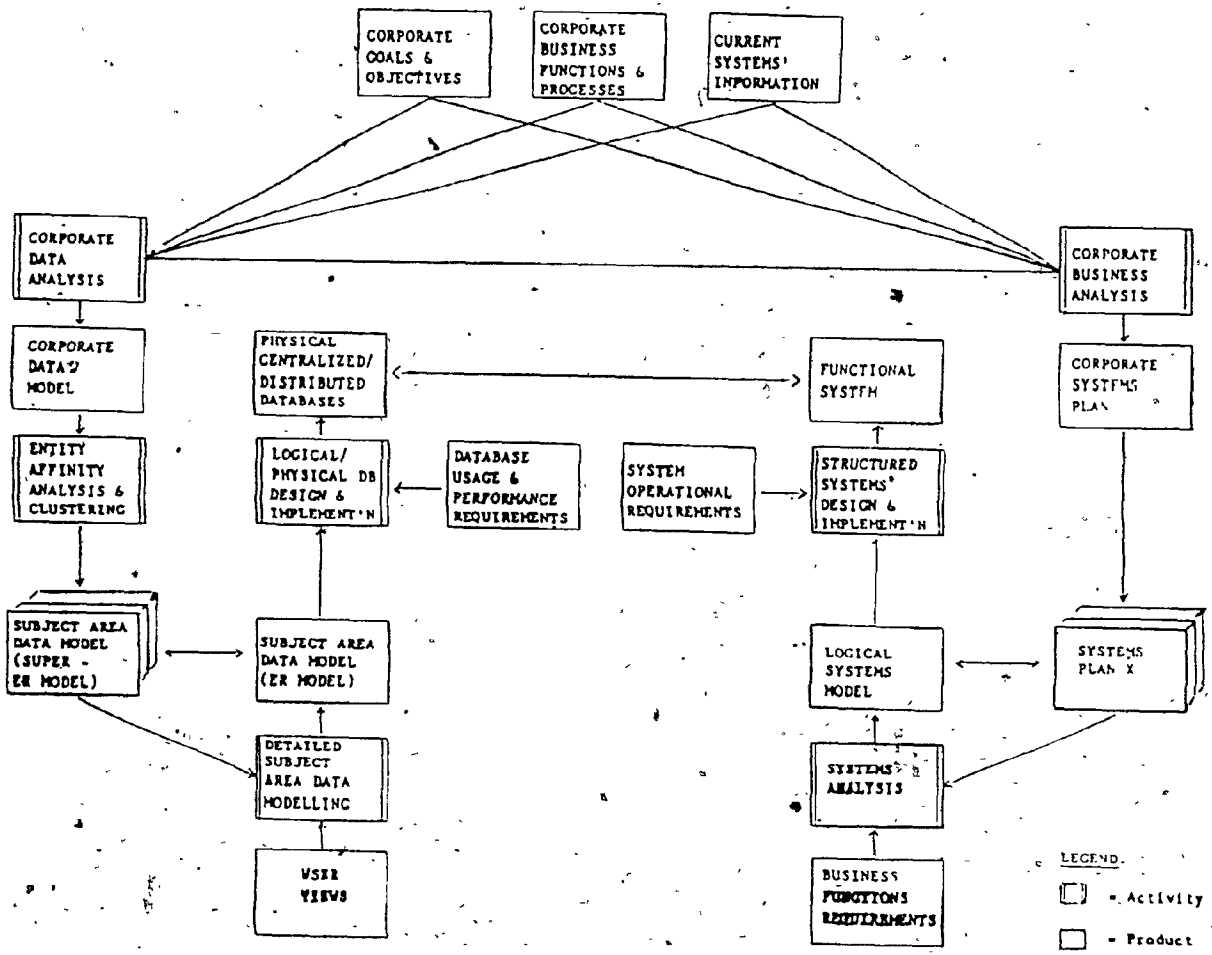


FIGURE 2.12 TOP-DOWN PLANNING AND BOTTOM-UP DESIGN/IMPLEMENTATION

are no longer ad hoc implementations but are essential components of a corporate information architecture - this is a significant departure from traditional approaches and cause of much confusion.

2.3 DATA ADMINISTRATION TOOLS

The success of the data administration function depends on the availability of a number of mechanized/non-mechanized tools or techniques. A number of manual/automated tools and/or techniques are currently available for systems and database development [Eastman,1975; Chen,1979b; Berild,1977; Gambino,1977; Bubenko,1978; Iossiphidis,1979; Newman,1982]. The exploitation of DA tools and techniques has a direct impact on all phases of data management and control. The tools can be grouped into eight broad categories:

- o data base management systems
- o data dictionary/directory systems
- o data base design tools
- o file design tools
- o application generators
- o utilities
- o project management systems
- o manual techniques

2.3.1 DATA ADMINISTRATION TOOLS - A SURVEY

DATA BASE MANAGEMENT SYSTEMS

Data base management system (DBMS) is a software tool for the physical management of data. Presently, there exist several commercial implementations of DBMS - they all have the following features in common: integrated source of data for multiple users, easy access to data, facilities for storage of large volume of data, capabilities for data sharing, measures for security and privacy of data, data integrity-constraints, and data independence [Zimmermann,1975; Chamberlin,1976; Taylor,1976; Tsichritzis,1976; Date,1977; Lawrence,1979; Sandberg,1981].

DATA DICTIONARY/DIRECTORY SYSTEMS

Data Dictionary/Directory Systems (DD/DS) is a software tool that aids the collection, storage and dissemination of data about data (i.e., meta-data) as a logically controlled, and standardized utility function [Ehrensberger,1976; Lefkovits,1977; NBS,1978; Ross,1981]. There exist several commercial implementations of DD/DS - the features offered by DD/DS include:

- o Unique identification, physical characteristics, and textual description for each of the data elements
- o Relationships of data elements to each other; and to components of system, e.g., to program, modules,

datastores, reports, etc.

- o Source, location, usage and destination of data - i.e., dataflow information
- o Validation and redundancy checking
- o Security/privacy measures
- o Query language and reporting capabilities
- o Facilities for interfacing with DBMS

DATA BASE DESIGN TOOLS

Data base design is a complex process. Since database design involves mapping parts of reality as viewed by human users to artificial structures, the human element is a necessary part of the design process [Bubenko,1978]. However, there exist several areas where tools can play an important role. IBM's Data Base Design Aid (DBDA) [IBM, 1975] is an example of a representative database design tool. DBDA automates significant part of the database design process - the design facilities that are supported include: gathering of requirements, creation of a structural model, physical design parameters selection, and design documentation. Several software tools have been proposed/implemented in recent years in one or more of the following DB design areas: logical DB design, physical DB design, design evaluation and prototyping, performance monitoring [Teichroew,1977; Piney,1977; Berild,1977; Roussopoulos,1979; Chan,1979; Irani,1979; Liberman,1979;

Martin,1981; Clemons,1981]

FILE DESIGN TOOLS

The file design tools support the traditional file design tasks - file organization selection, simulation, update and retrieval performance, file conversion, etc. [Mitoma,1975; Nunamaker,1976; Chen & Yao,1979b].

APPLICATION PROGRAM GENERATORS

The application program generators typically consist of a number of functional modules of generalized code, which can be readily selected and tied together to meet the needs of a particular application. These modules carryout the 'run-of-the-mill' functions common to all applications. Only code specific to the particular application need then be written [Kebel,1980; Bertrand,1980].

DATA LANGUAGES

This tool provides capabilities for data design. In particular, the facilities include language tools for gathering, editing and validating userviews, defining relations, describing data models, etc. [Machgeels,1976; Konsynski,1979].

QUERY LANGUAGES

Query languages are a tool which provide capabilities to the user to extract information from databases, i.e., languages such as QBE, SQL, MAPPER [Chamberlin,1976a,b; Zloof,1977; Univac,1980b; Codd,1982a,b]. To date, significant research has been carried out on query languages to enhance their ease of use. The new generation of query languages have English-like syntax, they are easy to use, and they need very little user training.

UTILITIES

Several software tools can be named that are pertinent to DA functional areas. These include text editors, data validators/auditors, text/file formatters, data entry systems, information retrieval systems, report generators, crossreference/KWIC index generators, data auditors, test data generators, flowchart/dataflowgram generators, prototyping aids, data modelling aids, etc. [Teichroew,1971; Newman,1982].

PROJECT MANAGEMENT SYSTEMS

The tools in this category address the administrative side of the systems and database development tasks. The facilities offered by the project management systems include activity scheduling, activity monitoring, activity control,

etc. [Univac,1980a].

MANUAL TOOLS/TECHNIQUES

A number of manual tools/techniques can aid the DA function. These tools/techniques are productivity aids and address such diverse areas as business analysis, systems design, systems documentation, project management, project reviews, etc. The applicable tools/techniques are:

- o CPM/PERT Charts
- o GANTT Charts
- o Data Matrices
- o HIPO (Hierarchy plus Input-Process-Output)
- o Petri Networks
- o Warnier-Orr Charts
- o Nassi-Schneiderman Charts
- o Flowcharts
- o Dataflow Charts
- o Structured English
- o Decision Tables
- o Walkthroughs
- o Technical Reviews, etc.

2.3.2 DATA ADMINISTRATION TOOLS - PROBLEMS AND ISSUES

One of the major problems encountered with software tools is assembling of diverse software packages in a single environment [Teory,1978; Howden,1982]. Presently, the DA

tools by and large are ad hoc implementations, each having its own idiosyncracies. Matching and adapting these tools to a DA environment needs considerable forethought, investment and planning. An integrated data administration support system can alleviate this problem. Such a system, can provide a comprehensive, convenient and friendly environment for all users involved with data administration. A system of this nature is designed and a major segment of it is implemented. The system is called DASS (Data Administration Support System) and is described in the remaining Chapters.

CHAPTER 3

DASS - CONCEPTS AND FACILITIES

3.1 DASS DESIGN OVERVIEW

3.1.1 DASS DESIGN PHILOSOPHY

At present, data administration support facilities are offered as standalone products or as extensions to the data dictionary and/or data base management systems. The disadvantage of the former is a proliferation of support systems, while that of the latter is a tampering of true roles of the data dictionary and data base management systems. A more practical approach is to create an integrated support system, which operates at a global level and provides controlled access to all facilities within the realm of data administration. The benefits of such an approach are:

- o centralized control of data
- o ease of conceptualization of data administration functions, their dependencies and interrelationships
- o an user friendly environment for the entire data administration community - data administrator, records administrator, database administrator, file/database designers, data analysts and end-users
- o ease of provision of training

- o a logical framework for the evolution and support of data administration facilities
- o ease of integration of newer data administration support facilities
- o ease of replacement of existing data administration facilities by newer or more efficient facilities
- o ease of maintenance of system-wide compatibility between component facilities
- o ease of implementation of security, privacy and integrity controls
- o efficient usage of computer systems resources
- o minimal maintenance and support costs
- o controlled growth of data administration facilities

The realization of the above listed benefits has been a primary motivation behind DASS - Data Administration Support System, proposed in this thesis. A detailed description of DASS is provided in the the sections that follow.

3.1.2 DASS STRUCTURE - AN OVERVIEW

The Data Administration Support System is an integrated facility that provides all the tools required for data administration. The basic components of DASS are:

- o an interactive driver program
- o a 'menu' handler.

- o a non-procedural data design language
- o a control command language
- o a meta-data database
- o a control database, and
- o a help and training facility

Figure 3.1 describes the modular structure of DASS. Each module shown in Figure 3.1 represents a functional sub-system of DASS. At a DASS session, the selection of a sub-system and the selection of a task within the selected sub-system are controlled through menus. The execution of the selected task is controlled by the user through the non-procedural, interactive data language facility.

3.1.3 DASS SUB-SYSTEM DESCRIPTION

ENTERPRISE DATA MANAGER

This sub-system provides a capability to the data administrator to define the enterprise profile and to monitor the enterprise from a data administration point of view.

DATA SECURITY/PRIVACY MANAGER

This sub-system is a tool for the data administrator to define and control the security/privacy of data stored in the meta-data database. The data security/privacy issue is

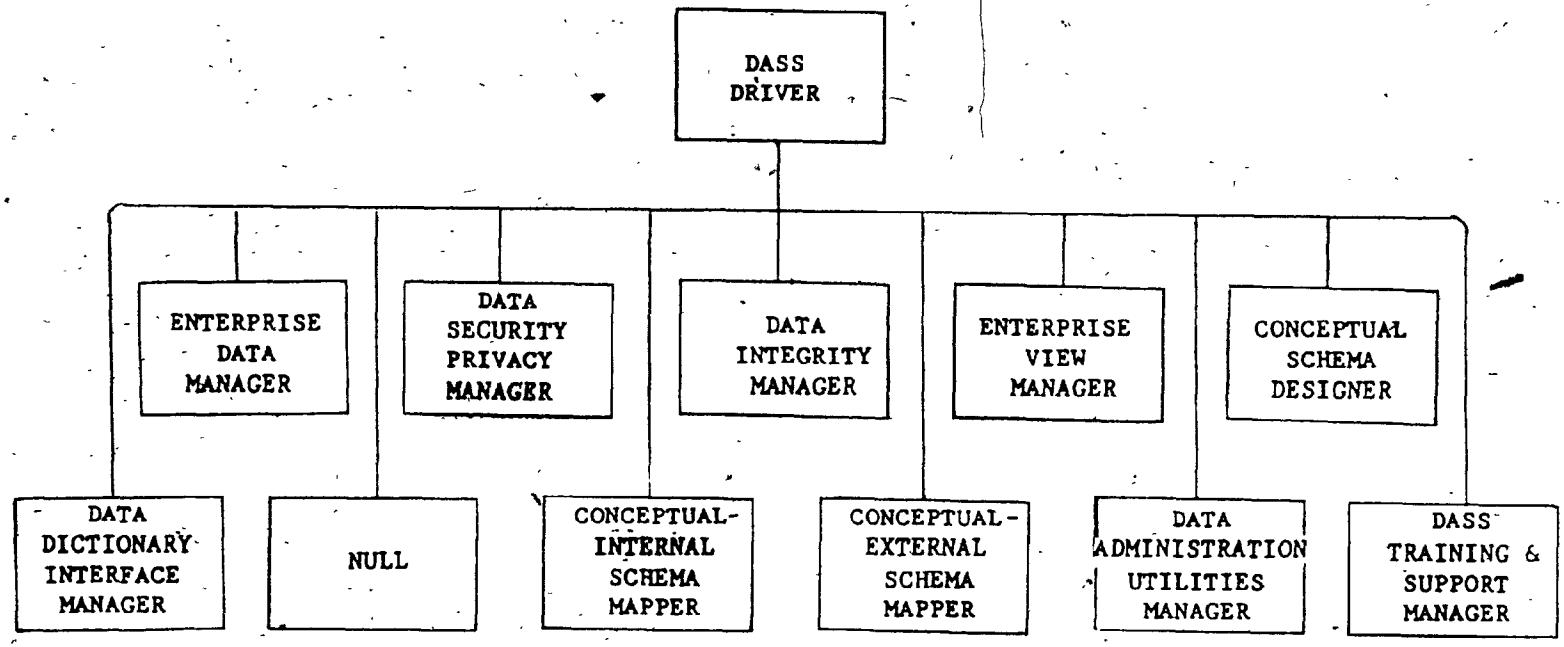


FIGURE 3.1 DATA ADMINISTRATION SUPPORT SYSTEM STRUCTURE

prime in DASS - this is because the background database is potentially accessible to the entire data administration community. Also, this sub-system facilitates definition of 'security/privacy types' that are referenced in the definitions of entities, relationships, attributes, etc.

DATA INTEGRITY MANAGER

This sub-system is a tool for the data administrator to ensure integrity of data in the meta-data database. Data integrity is essential as the 'producers' of meta-data are not necessarily the 'consumers' of meta-data. Also, this sub-system facilitates definition of 'integrity types' that are referenced in the definition of entities, relationships, attributes, etc.

CONCEPTUAL SCHEMA DESIGNER

The conceptual schema designer sub-system is the prime tool of DASS. Its target user community includes the data administrator, records administrator file/database designers, data analysts and end-users. The subsystem supports a variety of processes related to the conceptual schema component definition, schema generation, verification and validation.

ENTERPRISE VIEW MANAGER

This sub-system provides various views of the enterprise in the ongoing data administration process. The intended users for this facility include the entire data administration community. In the traditional analysis, modelling and design process, we use matrices, lists, reports, etc., to describe an enterprise. While these tools are quite satisfactory, they do not provide ease for conceptualization. This is because the matrices, lists or reports that describe an enterprise are a two dimensional projection of a multi-dimensional reality. The enterprise view manager sub-system facilitates obtaining a far richer view of an enterprise through the use of computer graphics, enterprise view diagrams (EVDs), and stepwise view refinement techniques. The stepwise view refinement consists of global, contextual and local level abstractions of the enterprise. Enterprise viewing is more fully described in section 3.4.

DATA DICTIONARY INTERFACE MANAGER

The data dictionary interface sub-system is intended to be a means to provide controlled access to the data dictionary. The interface is essential for the movement of data between the dictionary and the DASS meta-data database.

CONCEPTUAL-INTERNAL SCHEMA MAPPER

In the three-schema database architecture [ANSI,1975], there is a need for a mechanism to perform conceptual-internal schema mappings. This sub-system provides facilities to manage the software/hardware parameters that are an ingredient to the map, and to perform the actual mapping. The mapping is the responsibility of the database administrator.

CONCEPTUAL-EXTERNAL SCHEMA MAPPER

This sub-system is similar to conceptual-internal schema mapper. The difference is that the facilities are geared for conceptual-external schema mapping. The choice of this mapping is the responsibility of the database administrator.

DATA ADMINISTRATION UTILITIES MANAGER

A number of utilities are required to support data administration tasks. These utilities include editors, text formatters, report generators, matrix generators, information retrieval aids, cross-reference generators, sort/merge aids, picture processors, statistical analysis tools, etc. The data administration utilities sub-system provides a single entry point for commonly used utilities. The target users of this sub-system include the entire data

administration community.

TRAINING AND SUPPORT MANAGER

The training and support sub-system is an enhanced 'HELP' feature. The facilities provided by the sub-system include: DASS training, error diagnostic and command syntax description, and DASS operational environment control. The training and support facility makes it possible to use DASS without much prior training.

3.2 DASS DATA DESIGN LANGUAGE

The data design language offered by DASS is a composite of two languages: a text oriented data design language and a picture oriented data design language. Since, the conceptual schema designer facility is used by a broad spectrum of users, the data design language needs to be human-engineered, i.e., it should exhibit certain characteristics not generally found in problem oriented languages. Some of these characteristics are:

- o Simplicity of Syntax - the syntax should be simple. A cryptic syntax might be appealing to a technically oriented person but may not provide the ease of usage to an end-user
- o Flexibility - the language should be flexible, i.e. the language should not be controlled by a rigid syntax. The language should support multiple forms

of input: sentential, query-answer, query-by-example, tabular, etc.

- o Resemblance to Natural Language - the language should resemble a natural language. Such resemblance promotes ease of usage and accelerated learning. Further, it should exhibit some artificial intelligence capabilities i.e. minor syntax variations or omissions should not cause major semantic implications
- o User-friendly Interface - the language should support meaningful dialogue between the user and the machine. Novice users are unimpressed with CPU speeds, disk storage capabilities, or elegant file structures. For them the system appears the way it interacts when they make a mistake or are unsure about what to do next [Schneiderman,1982].

The importance of addressing human-engineering concerns in the design of man-machine interfaces has been stressed by several authors [Ling,1980; Ehrich,1982; Stamen,1982]. The DASS data design language has been designed with these concerns in mind.

3.2.1 DATA DESIGN LANGUAGE FEATURES

The DASS data design language has the following characteristics:

SYNTAX

In order to support a diverse class of users, the DASS syntax has been designed to adapt to different user styles and preferences.

Example:

To add an attribute 'beta' to a previously defined entity, say 'alpha', any of the following forms of commands are acceptable to the conceptual schema designer subsystem of DASS.

- o ADD ATTRIBUTE(beta) to ENTITY(alpha)
- o ADD A(beta) to E(alpha)
- o ADD A,beta to E,alpha
- o ADD ATTR 'beta' to ENTT 'alpha'
- o ADD ATTR/beta to ENTT/alpha

FLEXIBILITY

The language is flexible and user friendly. If a command is incomplete, DASS enters a conversational build mode and elicts missing components of the command, part by part. The language also supports tabular command input mode - here, components of a command are identified by column titles and user is solicited to enter command components under each column header.

RESEMBLANCE TO NATURAL LANGUAGE

The DASS data design language is English-like. It can correctly interpret commands with minor syntax variations and/or omissions.

Example:

While drawing an ERD or EVD, to add a line between two blocks numbered 11 and 12, any of the following forms of commands are acceptable to DASS.

- o ADD a LINE BETWEEN BLOCKS 11 and 12
- o ADD LINE 11,12
- o ADD LINE at 11,12
- o JOIN BLOCKS 11 and 12
- o JOIN BLOCKS 11 & 12
- o JOIN BLOCKS NUMBERED 11 and 12
- o CONNECT BLOCKS 11 & 12
- o CONNECT 11,12
- o DRAW a LINE at 11,12

  etc.

As it can be noticed, our DASS offers a command verb substitution capability. This capability is supported by maintaining a synonyms table for command verb keywords. Upon command synonym interpretation, DASS verifies with the user the accuracy of the interpretation by displaying the command and soliciting a 'go' or 'nogo'. This verification

step can be by-passed, if desired, by using the ACOM feature discussed in Chapter 5 (Section 5.5.1).

BNF NOTATION

The DASS data design language syntax can be expressed in Bakus-Naur Form (BNF) as follows:

<command> ::= <commandverb> <whatcomponent>
<wherecomponent> <howcomponent>

Example:

The command to rename an attribute of an entity is of the form:

RENAME ATTRIBUTE(alpha) of ENTITY(beta) as ATTRIBUTE(gamma)

Here:

<commandverb> is RENAME

<whatcomponent> is ATTRIBUTE(alpha)

<wherecomponent> is ENTITY(beta)

<howcomponent> is ATTRIBUTE(gamma).

3.2.2 DATA DESIGN LANGUAGE COMMAND SET

TEXT ORIENTED DATA DESIGN LANGUAGE COMPONENT

The text oriented data design language component has the following command set.

ADD commands:

The following commands are available for adding various text blocks to the meta-data database.

ADD gl(I1)

ADD al(I1) el(I2)

ADD vl(I1) ul(I2)

ADD pl(I1) fl(I2)

ADD tl(I1) pl(I2)

DELETE commands:

The following commands are available for deleting various text blocks from the meta-data database.

DEL gl(I1)

DEL al(I1) el(I2)

DEL vl(I1) ul(I2)

DEL pl(I1) fl(I2)

DEL tl(I1) pl(I2)

CHANGE commands:

The following commands are available for changing various text blocks in the meta-data database.

CHA gl(I1)

CHA al(I1) el(I2)

```
CHA v1(I1) u1(I2)
CHA p1(I1) f1(I2)
CHA t1(I1) p1(I2)
```

MOVE commands:

The following commands are available for moving an attribute from one entity to another, an userview from one user to another, etc.

```
MOV a1(I1) e1(I2),e2(I3)
MOV v1(I1) u1(I2),u1(I3)
MOV p1(I1) f1(I2),f1(I3)
MOV t1(I1) p1(I2),p1(I3)
```

LIST commands:

The following commands are available for listing various text blocks from the meta-data database.

```
LIS g1(I1)
LIS a1(I1) e1(I2)
LIS v1(I1) u1(I2)
LIS p1(I1) f1(I2)
LIS t1(I1) p1(I2)
```

SPLIT & MERGE commands:

The following commands are available for splitting and merging entities or relationships.

SPL e1(I1) e1(I2,I3)

MER e1(I1,I2) e1(I3)

RENAME commands:

The following commands are available for renaming various text blocks in the meta-data database.

REN g1(I1) g1(I2)

REN a1(I1) e1(I2) a1(I3)

REN v1(I1) u1(I2) v1(I3)

REN p1(I1) f1(I2),p1(I3)

REN t1(I1) p1(I2),t1(I3)

COUNT commands:

The following commands are available for counting various text blocks in the meta-data database.

COU g1

COU a1 e1(I2)

COU v1 u1(I2)

COU p1(I1) f1(I2)

COU t1(I1) p1(I2)

PROMOTE & DEMOTE commands:

The following commands are available for promoting an attribute; demoting an entity or relationship.

PRO a1(I1) e1(I2) e2(I3)

DEM e1(I1) a1(I2) e2(I3)

Legend:

A = {attribute}

E = {entity,relationship}

G = {enterprise,valueset,privacy,integrity,function,user}

U = {user}

V = {view}

F = {function}

P = {process}

T = {activity}

Ii $\in$ {x|x is an identifier}

ai $\in$ A; ei $\in$ E; gi $\in$ G; ui $\in$ U; vi $\in$ V

i $\in$ {1,2,3}

Note:

An identifier is a character string composed of letters and/or digits such that the leading character of the string is a letter, and the string is atmost six characters long.

The text oriented data design language commands permit a variety of syntax variations - see details under SYNTAX in Section 3.2.1.

PICTURE ORIENTED DATA DESIGN LANGUAGE COMPONENT

The picture oriented data design language component of DASS has the following command set.

ADD commands:

The following commands are available for adding various picture elements to an ERD/EVD.

ADD e1(I1) n1

ADD l1 n1,n2

ADD t1(I1) n1

ADD d1(I1,n1)

Note:

The picture elements correspond to the symbols of the ERD/EVD.

DELETE commands:

The following commands are available for deleting various picture elements from an ERD/EVD.

DEL e1(I1) n1

DEL l1 n1,n2

DEL t1 n1

DEL d1(I1,n1)

CHANGE commands:

The following commands are available for changing various picture elements in an ERD/EVD.

CHA e1(I1) n1 e2(I2)

CHA t1 n1 t2(I2)

MOVE commands:

The following commands are available for moving various picture elements in an ERD/EVD from one location to another.

MOV e1(I1) n1,n2

MOV l1 n1,n2 n3,n4

MOV t1 n1,n2

DISPLAY commands:

The following commands are available for displaying ERD's or EVD's.

DIS

DIS d1

DIS d1 w1(n1)

SPLIT & MERGE commands:

The following commands are available for splitting or merging various picture elements in an ERD/EVD.

SPL el(I1) n1,n2

MER el(I1,I2) n1

GET & PUT commands:

The following commands are available for retrieving and/or storing ERD's or EVD's.

GET d1(I1,n1)

GET d1(h1,n1)

PUT d1(I1,n1)

PUT d1(h1,n1)

ERASE commands:

The following commands are available for erasing the graphics screen.

ERA

ERA w1(n1)

GRID commands:

The following commands are available for displaying a grid on the graphics screen to facilitate I/O.

GRI

GRI w1(n1)

SCALE commands:

The following commands are available for displaying a scale along left and top edges of the screen to facilitate I/O.

SCA

SCA w1(n1)

LABEL commands:

The following commands are available for labelling various picture elements to facilitate ADD, CHANGE, or MOVE operations.

LAB

LAB w1(n1)

FRAME commands:

The following commands are available for drawing a frame around EVD's (see Section 3.4).

FRA

FRA w1(n1)

Legend:

D = {ERD,EVD}

E = {entity,relationship}

H = {hold}

L = {line}

T = {text}

W = {window}

Ii∈ {x|x is an identifier}

ni∈ {x|x is an integer}

i ∈ {1,2,3}

di∈ D; ei ∈E; hi∈ H; li∈L; ti∈T; wi∈W

Note:

See previous section for definition of the term 'identifier'.

The picture oriented data design language commands permit a variety of syntax variations - see details under SYNTAX in Section 3.2.1.

3.3 DASS CONTROL COMMAND LANGUAGE

The control commands provide a means for temporarily suspending an active task in lieu of accomplishing an utility function. They are identified by a '$' symbol preceding the command verb.

The control command syntax is as follows:

$<commandverb>,<commandoption> <commandspec>

In the current implementation of DASS, only 'HELP', 'PRINT', 'MOVE', 'CLEAR', 'SAVE', 'RESTORE', 'DATA' and 'SET' commands use 'commandoption' and/or 'commandspec'.

The control commands are of two types: global control commands and local control commands.

3.3.1 GLOBAL CONTROL COMMANDS

DASS provides fifteen global control commands. These control commands have no contextual requirements and have system-wide acceptance. The global control commands along with their meanings are given below:

- $ABORT - same as $STOP
- $CLEAR - clear text or display buffer
- $DATE - display current date
- $DISPLA - display ERD/EVD (using data in the display buffer)
- $HELP - enter training & support sub-system
- $MOVE - move text buffer contents to display buffer and vice versa
- $PRINT - print contents of text or display buffer
- $RESET - reset operating environment control flags
- $RESTOR - restore text or display buffer (see $SAVE)
- $SAVE - save contents of text or display buffer
- $SET - set operating environment control option (e.g., PLOT, DBUG, TEST, QUIK, etc.)

$STATUS - display operating environment control option status

$STOP - abnormally terminate processing and exit from DASS

$TIME - display clock time

$USAGE - Display accounting information

3.3.2 LOCAL CONTROL COMMANDS

The local control commands are similar to the global commands but have certain contextual requirements. For instance, the '$MENU' command works correctly if input at a menu display point, at other points, it creates an error condition and generates an error message. DASS provides eleven local control commands. The local control commands along with their meanings are given below:

$CURR - print current line from the active buffer

$DROP - same as $QUIT

$DATA - used in an input data stream to signal beginning of related data (see $END)

$END - used in the input data stream to signal end of related data (see $DATA)

$EXIT - exit from a conversational mode

$LIST - list contents of the current buffer

$MENU - display full menu (used under QUICK mode)

$NEXT - alter data solicitation or display sequence (forward move)

$PREV - alter data solicitation or display sequence (backward move)

$QUIT - ignore any outstanding/default updates and exit from conversational mode

$TAB - display tabs to facilitate user input

3.4 DASS ENTERPRISE VIEWING

3.4.1 ENTERPRISE VIEW MODELLING

An integral part of any strategic data planning methodology is enterprise analysis - a study of the functions, processes, activities, entities and relationships that make up an enterprise. The enterprise analysis is a lengthy, obscure, and often a tedious task. This is because the amount of complexity a human mind can cope with at any given time is considerably less than that which is embodied in the manifestation of an enterprise. Abstraction tools therefore play an important role in enterprise analysis. The enterprise data model (enterprise schema) is one such tool, and has been discussed in Chapter 2. Another useful tool is 'enterprise viewing facility'. Enterprise viewing is based on 'stepwise view refinement' technique, i.e., a top-down viewing of the enterprise in terms of its business functions, processes, activities, entities and relationships. The enterprise viewing facility supported by DASS is based on a proposed enterprise view model.

ENTERPRISE VIEW MODEL

As defined in the previous Chapter, an enterprise is a subset of the real world that is of interest to a group of people. We can assume that the real world is populated by M enterprises of interest to some N groups of people. We can denote the resulting enterprise and group sets as follows:

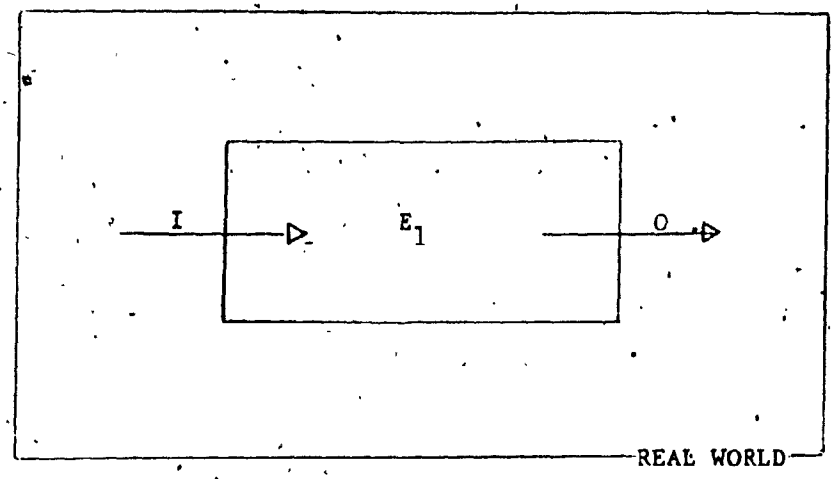
$\mathcal{E}$ = {Ei}, i=1,2,...,M

$\mathcal{G}$ = {Gi}, i=1,2,...,N

An enterprise interacts with the real world by producing products, services, actions, and accepting various inputs - see figure 3.2.

The real world sends to the enterprise far more information than it can profitably use. In this regard the enterprise is like a living organism which processes only a small portion of the data that impinges upon its senses. The type of interaction between the enterprise and the real world is largely determined by the products and services the enterprise produces.

Based on the enterprise profile discussed above we can propose an enterprise view model.



LEGEND:

E_1 = ENTERPRISE

I = INPUT STIMULI

O = OUTPUT STIMULI

FIGURE 3.2 AN ENTERPRISE

Enterprise View Model Definition:

An enterprise is a five-tuple E (I,O,P,M,T)

where:

E = enterprise,

I = input stimuli,

O = output stimuli,

P = a set of processes,

M = memory (data structure), and

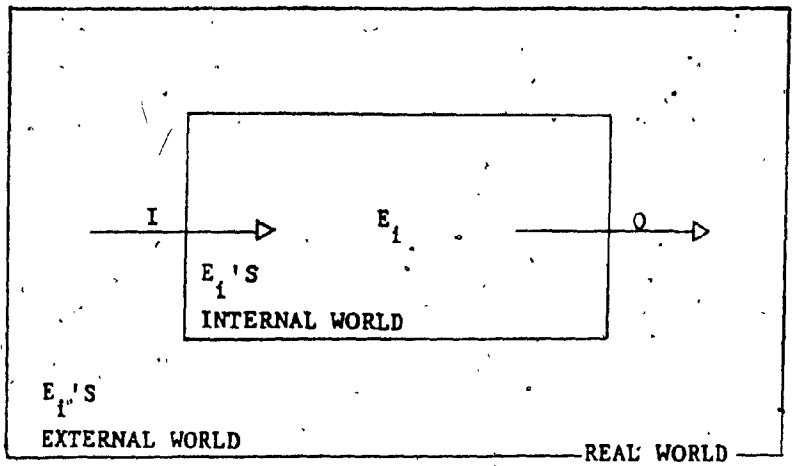
T = time.

Note:

The output stimuli $O(t)$ at time 't' is dependent on the present input $I(t)$ and also on the past inputs at instants $t-1, t-2, t-3, \ldots, t-n$. The dependence of the output on past inputs justifies the need for memory (M) and time (T) components. The memory is finite and corresponds to a set of internal states. At time 't', the enterprise 'E' is in some state M(to), it receives an input I(t1) and this present input and the previous internal states determine the next internal state M(t1).

Real World Partitioning:

An enterprise, as defined by the model, subsets the real world into two parts - we call these parts the enterprise world and the external world, respectively. The real world partitioning with respect to an enterprise is diagramatically described in figure 3.3.



LEGEND:

E_i = ENTERPRISE

I = INPUT STIMULI

O = OUTPUT STIMULI

FIGURE 3.3 REAL WORLD PARTITIONING WITH RESPECT TO ENTERPRISE E_i

Let 'ei' represent an enterprises in the real world. Let I(ei) and O(ei) denote the input and output stimuli of the enterprise ei. Then the input stimuli I(ei) of enterprise ei is given by:

$$I(ei) = \Delta + \sum_{j=1}^{n} O(ej); \text{ for all } j \neq i$$

Note:

Here, Δ stands for input from the real world to ei's external world.

Next, the enterprise ei itself can be considered as a composite of subenterprises, i.e., ei={eij}, j=1,2,...,n. In other words, we can consider the enterprise world as a real world (at a different conceptual level) and identify enterprises (or sub-enterprises) within it. Then the input stimuli I(eij) of subenterprise eij is given by:

$$I(eij) = I(ei) + \sum_{k=1}^{n} O(eik); \text{ for all } k \neq j$$

The conceptual level switching described above can be extended to any required number of enterprise levels

desired.

Enterprise View Model - Assumptions:

The enterprise view model proposed above is based on the following assumptions:

- o the input stimuli to a sub-enterprise is of the form 'a + b', where 'a' is the input stimuli from the external world, and 'b' represents the total output stimuli of those 'sub-enterprises' within the sub-enterprise under consideration
- o the output stimuli from the enterprise world is of the form A(b), where 'A' is the enterprise-to-external-world output defining operator, and 'b' is as described above
- o each process pi reckons (i.e., senses) 'I' as a series of discrete stimuli and responds to each of the stimuli. The response is an output stimuli and/or a change of memory. The output stimuli and/or memory state change induced by a process could be null

3.4.2 STEPWISE VIEW REFINEMENT

STEPWISE VIEW REFINEMENT CONCEPTS

The enterprise view model described in the previous

section can be conveniently used for structured viewing of the enterprise. The technique consists of successive iterations of two main steps:

- o step 1 - picture the current level of enterprise view as a composite of functions, processes, activities or entities/relationships
- o step 2 - select one of the functions, processes, activities or entities/relationships as a candidate for further viewing

Each iteration of step 1 and step 2 triggers a conceptual level switch and provides a refinement of the enterprise view. Dass enterprise view manager currently supports three conceptual levels - these are called global, contextual and local, respectively. The equations corresponding to input and output stimuli (see Real World Partitioning) provide a means for checking the input-process-output stability/validity of the enterprise (sub-enterprise) at each conceptual level.

STEPWISE VIEW REFINEMENT - EXAMPLE

The enterprise view manager of DASS provides facilities for enterprise analysis via stepwise view refinement. Upon selection of the enterprise view manager option, the user is prompted for enterprise name input so that required enterprise view diagrams (EVDs) could be

retrieved from the meta-data database. Once retrieved, the EVDs are displayed at respective conceptual levels.

Global Level:

This is the highest level of abstraction and the display consists of enterprise business functions (subenterprises). A typical display is shown in Figure 3.4.

Upon display of the global view, the user is prompted to select one of the subenterprises so that a refinement of that subenterprise can be provided at the next conceptual level.

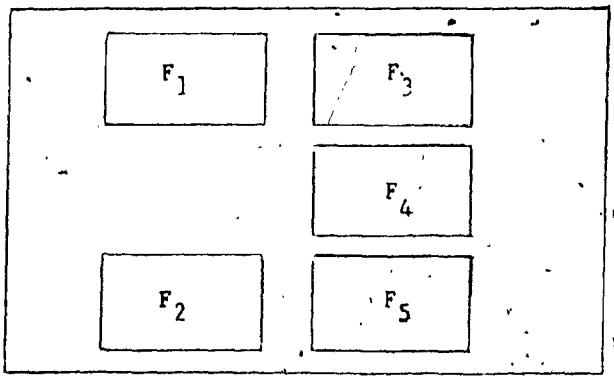
Contextual Level:

This level provides an intermediate level of abstraction of the enterprise. A typical contextual level display is shown in Figure 3.5.

Upon display of the contextual view, the user is prompted to select one of the business processes so that a refinement of the selected process can be provided at the next conceptual level.

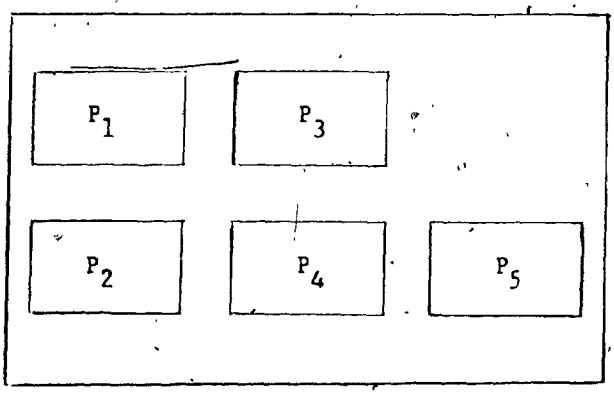
Local Level:

This is the lowest level of abstraction of the enterprise where the selected process is described in terms of activities and in terms of entities/relationships required to support the activities. The Figures 3.6 (a) and 3.6 (b) show the typical displays at the local level. Figure



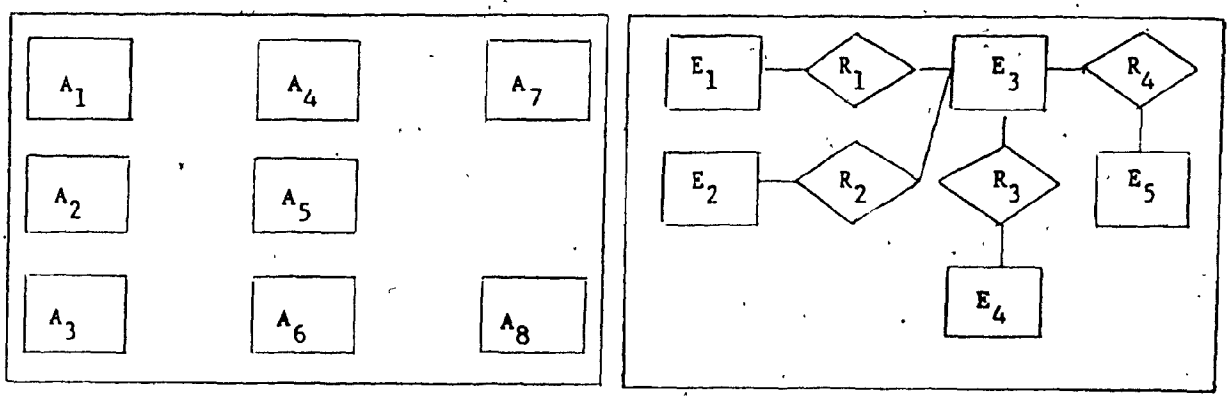
LEGEND: F_i ARE FUNCTIONS

FIGURE 3.4 ENTERPRISE VIEW (GLOBAL)



LEGEND: P_i ARE PROCESSES

FIGURE 3.5 ENTERPRISE VIEW (CONTEXTUAL)



(a) (b)

LEGEND: A_i, E_i AND R_i ARE ACTIVITIES, ENTITIES AND RELATIONSHIPS RESPECTIVELY

FIGURE 3.6 ENTERPRISE VIEW (LOCAL) - BUSINESS MODEL & E-R MODEL

3.6 (a) is a block diagram which shows the underlying business activities. Figure 3.6 (b) is an entity-relationship diagram which shows the entities and relationships that support the business activities. The enterprise viewer output from an actual DASS session is provided in Appendix 3.

3.5 DASS IMPLEMENTATION

Because of its enormous size, there is a need to implement DASS in phases. The current implementation of DASS comprises of:

- o an architectural framework within which each of the sub-systems described earlier (see Section 3.1.3) fit together
- o the required facilities for enterprise view definition and display
- o the required facilities for conceptual schema component definition, schema generation and verification
- o the required facilities for training & support

In the following description of DASS structure, an asterisk (*) besides a subsystem name or subsystem option name indicates that it is not implemented under the current version.

Based on its design/implementation, DASS processing is characterised by the following sequence of actions:

- o Invocation & Primary Menu Selection
- o Secondary Menu Selection
- o Command Input
- o Command Execution

The DASS primary and secondary menu hierarchy is described in Appendix 1. The DASS data design language commands and control commands are described in Sections 3.2 and 3.3, respectively.

3.5.1 DASS INVOCATION & PRIMARY MENU SELECTION

To invoke DASS, the user simply types-in the following command:

"CALL DASS"

On invocation, DASS responds by printing system identification information, date and time of call, and version/level details. The following primary menu is then displayed:

```
DASS SUB-SYSTEMS:
-----------------

01. Enterprise Data Manager
02. Data Security/Privacy Manager
03. Data Integrity Manager
04. Enterprise View Manager
```

05. Conceptual Schema Designer

06. Data Dictionary Interface Manager*

07. Conceptual-Internal Schema Mapper*

08. Conceptual-External Schema Mapper*

09. Data Administration Utilities Manager*

10. Training and Support Manager

<1> TYPE-IN SELECTION:

The primary menu, as it can be seen, offers ten options - DASS solicits a option selection number from the user via display of the message:

<N> TYPE-IN SELECTION:

The DASS set of menus form an onionshell-like structure. The menu selection process corresponds to moving from an outer layer to an inner layer and vice versa. The 'N' within triangular brackets above, represents the currently active layer. The options to select here are 1,2,3,4,5 or 10. Other options are not supported in the current implementation of DASS. However, selection of a non-supported option does not cause an error termination - DASS will simply display a diagnostic message and re-solicit option selection.

3.5.2 DASS SECONDARY MENU SELECTION & PROCESSING

The selection of a subsystem under the primary menu always triggers display of a secondary menu. The secodary menu that is displayed is unique to the subsystem selected. The following sections describe the secondary menu options and processing facilities provided under each subsystem.

ENTERPRISE DATA MANAGER SUBSYSTEM

The enterprise data manager sub-system offers the following secondary menu:

```
ENTERPRISE DATA MANAGER SUBSYSTEM:
----------------------------------
01. Enterprise Profile Definer
02. Enterprise Data Monitor*
<2> TYPE-IN SELECTION:
```

Upon selection of the Enterprise Profile Definer option, DASS prompts the user for command input. The Enterprise Profile Definer option permits definition of the enterprise profile. The responsibility for the enterprise definition falls on the data administrator. The valid commands under this option are:

- o ADD ENTERPRISE (enterprisename)
- o DELETE ENTERPRISE (enterprisename)
- o CHANGE ENTERPRISE (enterprisename)
- o LIST ENTERPRISE (enterprisename)
- o RENAME ENTERPRISE (enterprisename1)

as ENTERPRISE (enterprisename2)

Note:

All data design language command mnemonics and keywords can be abbreviated to three characters.

Example:

The last command in the above list can be entered as follows:

REN ENT(enterprisename1) as ENT(enterprisename2)

The initial definition of enterprise is performed in conversational mode using a pre-defined query profile. The pre-defined query profile for enterprise definition is shown below:

Enterprise Query Profile:

- o Enterprise Name ?
- o Enterprise Synonym ?
- o Security Password ?
- o Enterprise-ID/Location ?
- o Enterprise Description/Definition (line 1) ?
- o Enterprise Description/Definition (line 2) ?
- o Enterprise Description/Definition (line 3) ?

The query profiles used by data security/privacy manager and data integrity manager, are similarly pre-defined. Once the

enterprise profile has been defined, user profiles, functions, entities, relationships and valuesets under the enterprise can be defined.

DATA SECURITY/PRIVACY MANAGER SUB-SYSTEM

The data security/privacy manager sub-system offers the following secondary menu:

```
DATA SECURITY/PRIVACY MANAGER SUBSYSTEM:
----------------------------------------
01. Data security/privacy Definer
02. Data security/privacy controller*
<2> TYPE-IN SELECTION:
```

Upon selection of the Data Security/Privacy Definer option, DASS prompts the user for command input. The Secutity/Privacy Definer option provides facilities to define the security/privacy types for the information existing under the enterprise. The commands for security/privacy definition can be derived from the enterprise definition commands by substituting 'PRIVACY' for 'ENTERPRISE'.

The initial definition of a security/privacy type is accomplished in conversational mode using the pre-defined query profile. Once a security/privacy type has been defined, it can be referenced in the definitions of

entities, relationships, etc.

DATA INTEGRITY MANAGER SUB-SYSTEM

The data integrity manager sub-system offers the following secondary menu:

DATA INTEGRITY MANAGER SUBSYSTEM:

---

01. Data Integrity-constraint Definer

02. Data Integrity-constraint Monitor*

<2> TYPE-IN SELECTION:

Upon selection of the Data Integrity-constraint Definer option, DASS prompts the user for command input. The Integrity-Constraint Defintion option facilitates definition of integrity-constraint types with respect to entities and relationships. The integrity-constraint definition format supported by DASS is similar to those described by Bubenko [Bubenko,1980], Tsichritzis [Tsichritzis,1982] and Machgeels [Machgeels,1976].

Example:

The integrity-constraint which asserts that an employee's monthly salary must not decrease can be formulated as follows:

$$\text{EMPSAL } [\forall m \forall e (\forall t(\text{employee}(e,t) \wedge \text{month}(t) = m) \wedge \forall t(\text{employee}(e,t) \wedge \text{month}(t) = m-1) \wedge (\text{salary}(e,m) > \text{salary}(e,m-1)))]$$

Here, e, m, and t stand for an employee occurrence, a month occurrence, and a time instance. For the integrity-constraint to hold, it is required by the above assertion that the employee is fully employed for two consecutive months.

The commands for integrity-constraint definition can be derived from the enterprise definition commands by substituting 'INTEGRITY-CONSTRAINT' for 'ENTERPRISE'.

The initial definition of an integrity-constraint type is accomplished in conversational mode using the pre-defined query profile. Once the integrity-constraint has been defined, it can be used in the definition of entities, relationships, etc.

ENTERPRISE VIEW MANAGER SUB-SYSTEM

The enterprise view manager sub-system offers the following secondary menu:

```
ENTERPRISE VIEW MANAGER SUBSYSTEM:
----------------------------------
01. Enterprise View Definer
02. Enterprise View Displayer
<2> TYPE-IN SELECTION:
```

The Enterprise View Definer (EVD) offers two options, viz., EVD Formatter option and EVD Utility Controller option. Under EVD Formatter option, the user can format enterprise view diagrams (i.e., add, change, delete, and display EVD's). Under EVD utility Controller option, the user is provided with facilities for direct creation of EVD display files, loading of off-line coded EVD display files, and definition of enterprise view charts. An enterprise view chart is a composite of EVD's defined for graphics plotter output.

The Enterprise View Displayer provides a capability for viewing the enterprise. The concepts and scope of enterprise viewing has been discussed in section 3.4. The enterprise view displays from an actual DASS session are provided in Appendix 3.

CONCEPTUAL SCHEMA DESIGNER SUBSYSTEM

The conceptual schema designer sub-system offers the following secondary menu:

CONCEPTUAL SCHEMA DESIGNER SUBSYSTEM:

---

01. Conceptual Schema Component Definer
02. Conceptual Schema Generator
03. Conceptual Schema Viewer

04. Conceptual Schema Validator*

<2> TYPE-IN SELECTION:

The conceptual schema component definer provides facilities to build (i.e., add, change, delete and list) user profiles, userviews, functions, valuesets, entities, relationships and attributes.

The conceptual schema generator offers facilitiies to define the conceptual schema (i.e., enterprise schema) in terms of an entity-relationship diagram (ERD).

The conceptual schema viewer offers facilities for partial/total viewing of the conceptual schema. The views offered are: entity view, relationship view, attribute view, entity-relationship structure view, and the total schema (ERD) view.

The Conceptual Schema Designer conepts and facilities are discussed in detail in Chapter 4.

TRAINING & SUPPORT MANAGER SUBSYSTEM

The training and support manager subsystem offers the following menu:

TRAINING AND SUPPORT MANAGER SUBSYSTEM:

---

01. Training Handler

02. Explanation Handler

03. Executive Request Handler

<2> TYPE-IN SELECTION:

The training and support manager implementation supports two modes of access to the subsystem: via menu selection or via the '$HELP' command.

The training handler option supports the training function. DASS training consists of several tutorial modules. Once in training mode, the user is provided with a forward and backward browse capability, which permits user control of the training pace and topic selection.

The explanation handler option supports two functions, viz., error code explanation, and command syntax description. The user provides DASS either an error code or a command mnemonic to obtain the desired information.

The executive request handler option supports three categories of functions, viz., operating environment control, trace/debug, and monitoring.

The operating environment control functions include:

ECHO - for echo control of user input

PLOT - for control of plot facilities access

ACOM - for assume command control

QUIK - for control of displayed information

The trace/debug functions include:

TEST - for test mode operation of DASS

DBUG - for debug mode operation of DASS

Under test mode all database updates are reckoned as being temporary, i.e., for the duration of the DASS session only. The DBUG mode provides 'trace' information corresponding to tasks executed during a DASS session, for use in system debugging. Both, 'TEST' and 'DBUG' functions are intended to be system maintenance oriented capabilities.

The monitoring functions include:

MCOM - for monitoring all DASS commands

MDBA - for monitoring database access

MGRA - for monitoring graphics facility access

The MCOM function permits monitoring of the execution of each DASS command. The MDBA function provides a capability to monitor database access. The MGRA function is similar to MDBA but is applicable to graphics facility access. The information provided by the above functions can be used for control of DASS command processing, database access tuning and graphics facility access optimization.

Note:

All executive request commands are enabled or disabled via the 'set' command. The syntax of the 'set' command has been discussed in section 3.3 under DASS Control Command Facility.

3.5.3 DASS SESSION TERMINATION

Two control commands are provided by DASS for process termination purposes:

EXIT - for normal termination of processing at layer 'n'

ABORT - for abnormal termination of a DASS session

As described earlier, DASS processing occurs at various layers. The effect of 'EXIT' at layer 'n' is normal termination of processing at layer 'n' and a pass of control to the (n-1)th layer. On the other hand, the effect of 'ABORT' at layer 'n' is abnormal termination of processing at layer 'n' and a pass of control out of DASS. A termination processing module is always invoked by DASS prior to a control flush. The termination processing ensures release of buffers, workfiles, databases and graphic facilities allocated internally by DASS. The termination processing also includes the display of accounting information. The accounting information comprises of statistical information gathered during the DASS session, which includes start/finish date and time, central processor

usage; exit type, number of database calls, number of graphic facility calls, and system usage charge.

CHAPTER 4

DASS - USER INTERFACE: CONCEPTUAL SCHEMA DESIGN

4.1 CONCEPTUAL SCHEMA DESIGN CONCEPTS

4.1.1 CONCEPTUAL SCHEMA - OVERVIEW

The conceptual schema is a description of an enterprise in terms of its inherent objects and relationships. In this regard, a conceptual schema is also called an enterprise schema. The meaning of the term 'enterprise' is contextual - it can stand for an entire organization, a division/department within the organization or simply a functional area. Under conceptual schema design, we attempt to model a part of reality, i.e., enterprise, in terms of its underlying objects and relationships. In order to do this we need a thorough understanding of the enterprise that we are trying to model [Tsichritzis,1975; Kent,1976; Biller,1977; Hsu,1979]. Consequently, the conceptual schema design process is preceded by a number of activities which help to understand the enterprise in question. These activities include:

- o Enterprise Definition
- o Security/Privacy Definitions
- o Integrity-constraint Definitions
- o Business Function/Process/Activity Definitions

- o User Profile Definitions
- o Userview definitions
- o Valueset definitions
- o Entity/Relationship Definitions
- o Entity/Relationship Arrtibute Definitions

Conceptual schema design, by nature, is creative and qualitative - for this reason it cannot be totally automated i.e., it pre-supposes presence of human element in its derivation [Senko,1977; Hubbard,1981]. The overall conceptual schema design process consists of gathering information about the enterprise, defining the conceptual schema components and storing the information in a meta-data database that aids easy storage, retrieval of stored information for the purpose of analysis, update of stored information upon refinement, and finally definition of a conceptual schema based on the captured information.

4.1.2 CONCEPTUAL SCHEMA DESIGN UNDER DASS

In the previous Chapter, the context of the conceptual schema design manager within DASS was explained. Further, Enterprise Definition, Data Security/Privacy Definition, Data Integrity-Constraint Definition procedures were discussed. These definitions are pre-conceptual Schema definition activities. In this section the facilities offered by the conceptual schema designer will be

investigated.

Upon selection of the conceptual schema designer sub-system under the primary menu, the the following secondary menu is displayed:

CONCEPTUAL SCHEMA DESIGNER SUBSYSTEM:

---

01. Conceptual Schema Component Definer

02. Conceptual Schema Generator

03. Conceptual Schema Viewer

04. Conceptual Schema Validator

<2> TYPE-IN SELECTION:

The strategy for conceptual schema design under DASS is as follows:

01. Define an enterprise and then proceed to define the security/privacy types, integrity-constraint types, user profiles, userviews, functions, processes, activities, entities, relationships, and attributes that are part of the enterprise using the text oriented data design language. Use the meta-data database to store the information

02. Retrieve, refine and re-store the defined information using the text oriented data design language

03. Identify entities and relationships using the top-down or bottom-up data analysis approach (see

note below)

04. Use conceptual schema generator to create a graphical representation of the enterprise, in terms of entities and relationships using the picture oriented data design language
05. Retrieve, refine and re-store the graphical representation of the enterprise using the picture orientd data design language
06. Use conceptual schema viewer to display various components of the conceptual schema, based on the textual and graphical data captured in steps 1 through 5. Also, use these perspectives to reiterate steps 1 through 5, above
07. Use conceptual schema validator to check consistency of the conceptual schema with respect to the information stored in the meta-data database

Note:

For identification of entities and relationships there exist two distinct approaches, viz., top-down and bottom-up. In the top-down data analysis approach, the information requirements at the activity level are first identified and then aggregated upwards to the function level. Based on the aggregate information requirements across functions, entities (i.e., super-entities) and relationships are identified for the enterprise under study. In the bottom-up data analysis appraoach, first, the userviews are gathered

and translated into local views; second, the local views are edited, analysed, and normalized to form 3NF relations; third, the 3NF relations from the previous step are synthesized to yield unique third-normal-form relations; and last, the unique 3NF relations are used as a basis to define the entities and relationships for the enterprise under study.

For developing a Conceptual Schema (i.e., Enterprise Schema) strictly for strategic planning purposes, the coarser top-down data analysis approach is most suitable. On the other hand, in the context of database design a fine-grained bottom-up data analysis approach is a necessity.

4.2 CONCEPTUAL SCHEMA DESIGN FACILITIES

4.2.1 CONCEPTUAL SCHEMA COMPONENT DEFINER

Upon selection of the conceptual schema component definer option the following menu is displayed:

```
CONCEPTUAL SCHEMA COMPONENT DEFINER:
-----------------------------------
01. User Profile Definer
02. Business Function/Process/Activity Definer
03. Userview Definer
04. Userview analyzer/synthesizer*
05. Valueset Definer
```

06. Entity/Relationship/Attribute Definer

<3> TYPE-IN SELECTION:

Out of the six options provided here, five are 'definition' options. The 'definition' options facilitate definition of user profiles, functions, processes, activities, userviews, valuesets, entities, relationships, and attributes. The definitions are performed in conversational mode using a pre-defined query profile. Each type of definition has its associated query profile. The query profile for entity definition is listed below:

ENTITY QUERY PROFILE:

- o Entity Name ?
- o Entity Synonym ?
- o Security Password ?
- o Entity-ID ?
- o Entity Description/Definition (line 1) ?
- o Entity Description/Definition (line 2) ?
- o Entity Description/Definition (line 3) ?
- o Entity Type (dependent/independent) ?
- o Entity Existence Probability ?
- o Entity Size ?
- o Entity Cardinality ?
- o Entity Expected Growth ?
- o Entity Access Frequency ?
- o Entity Access Volume ?

The conceptual schema designer uses the query profile to build the information block for the 'entity' being defined. Once the information block has been built it can be stored in the DASS meta-data database, retrieved, listed, updated, deleted, etc. The conversational definition mode offers flexibility - the user can re-initiate the conversation, move forward or backward in the query sequence, print the information block created upto the point, enter the training and support manager sub-system and then resume processing at the point of detour, take a pre-mature exit, etc.

In the following sections each of the conceptual schema component definer options is described in detail. The valid commands listed under each option have a myriad of variations - this flexibility of the DASS data design language has been discussed elsewhere (Chapter 3).

USER PROFILE DEFINER

This option provides facilities to define user profiles under an enterprise. The definition of users is a function of the enterprise administrator. The valid commands under this option are:

- o ADD USER (username)
- o DELETE USER (username)
- o CHANGE USER (username)

- o LIST USER (username)
- o RENAME USER (username1) as USER (username2)
- o COUNT USERS

BUSINESS FUNCTION/PROCESS/ACTIVITY DEFINER

This option provides facilities to define business functions, processes and activities under an enterprise. A business function is a composite of business processes, and each business process is a composite of business activities. The definition of functions, processes and activities is an end-user responsibility.

The commands for function definition can be derived from the user definition commands by substituting 'FUNCTION' for 'USER'.

The valid commands for process definition are:

- o ADD PROCESS (processname) to FUNCTION (functionname)
- o DELETE PROCESS (processname) of FUNCTION (functionname)
- o CHANGE PROCESS (processname) of FUNCTION (functionname)
- o LIST PROCESS (processname) of FUNCTION (functionname)
- o RENAME PROCESS (processname1) of FUNCTION (functionname) as PROCESS (processname2)
- o COUNT PROCESSSES of FUNCTION (functionname)

The commands for activity definition can be derived from the

process definition commands by substituting 'ACTIVITY' for 'PROCESS' and 'PROCESS' for 'FUNCTION'.

USERVIEW DEFINER

This option provides facilities to define userviews of the enterprise's data. The definition of userviews is an end-user function [Holland,1980; Finkelstein,1981]. The commands for userview definition can be derived from process definition commands by substituting 'USERVIEW' for 'PROCESS' and 'USER' for 'FUNCTION'.

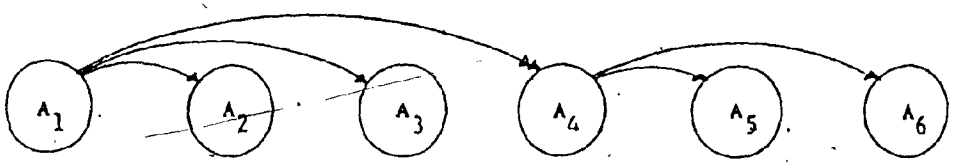
USERVIEW ANALYZER/SYNTHESIZER*

Once the user views have been defined, it is the data analyst's function to analyze the views for consistency/validity, and formally define the local views of the enterprise. A local view is a rigorous description of an userview comprising of graphical and textual parts - see Figure 4.1. The view analysis is followed by a normalization process. The objective here is to reduce relations identified in the previous step to Third-Normal-Form (3NF), so that the conceptual schema that is ultimately built depicts a stable model of the enterprise. To begin with, all local views are assumed to be un-normalized relations. These un-normalized relations are transformed into normal-form relations through decomposition - see Figure 4.2. The criteria applied for

o VIEW ID:
<id>

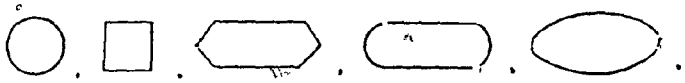
o VIEW NAME:
<name>

o VIEW POLYGRAPH:



<graphical part>

Note:
The nodes in the polygraph can be conveniently represented by any of the following symbols:



o VIEW DESCRIPTION:
<narrative>

o VIEW SOURCES:
<userview id's>

o VIEW SPECIFICATIONS:

data element	element type*	element size	value type	element sensitivity	integrity constraint
---	---	---	---	---	---
---	---	---	---	---	---
:	:	:	:	:	:

*key or non-key ?

o VIEW CONTEXT:
<batch/online?>

o VIEW VOLUME:
<physical instances/occurrences>

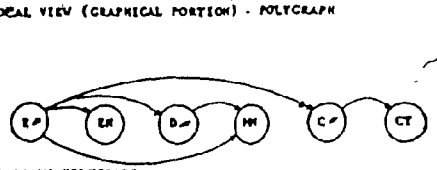
FIGURE 4.1 LOCAL VIEW PROFILE

- UN-NORMALIZED RELATION(USER VIEW)

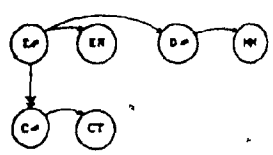
EMPLOYEE (employee#, employeename, department#, managername, course#, coursetitle)

Note:
employee# (E#) is the primary key, department# (D#) uniquely identifies managername (MN), course# (C#) uniquely identifies coursetitle (CT), and 'course#, coursetitle' forms a repeating group.

- LOCAL VIEW (GRAPHICAL PORTION) - POLYGRAPH

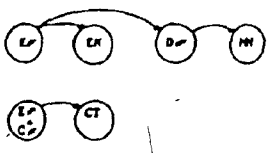


- RE-DRAWN POLYGRAPH



Note:
transitive dependency between E# & MN has been removed

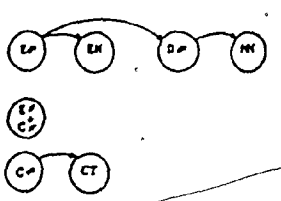
- NORMALIZATION - STEP 1



unchanged

1NF decomposition

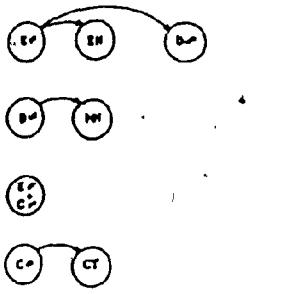
- NORMALIZATION - STEP 2



unchanged

2NF decomposition

- NORMALIZATION - STEP 3



3NF decomposition

unchanged

- NORMALIZED RELATIONS

The relations depicted in step 3 are all in 3NF. We can formally define these as follows:

EMPLOYEE (employee#, employeename, department#)

TRAINING (employee#, course#)

COURSE (course#, coursetitle)

DEPARTMENT (department#, managername)

FIGURE 4.2 NORMALIZATION PROCESS

first, second, and third normal form transformations are:

- o First Normal Form - the relation contains no repeating data groups
- o Second Normal Form - all non-key data items in the relation are fully functionally dependent on the primary key
- o Third Normal Form - all non-key data items in the relation are fully functionally dependent on the primary key and are independent of each other

Upon generation of 3NF relations corresponding to all local views, the relations are synthesized. The synthesis process ensures definition of unique set of third-normal-form relations for the enterprise under study.

VALUESET DEFINER

This option provides facilities to define valuesets within an enterprise. Recall that in the E-R model, each attribute is assigned a valueset. The valueset defines the range and type of values an entity could assume. For instance, we can have the attribute of an entity called car assigned to a valueset called carcolour, e.g., valueset carcolour = {blue, green, yellow, orange, red}. The valueset definition format supported by DASS is similar to those described by Bubenko [Bubenko,1980] and Tsichritzis [Tsichritzis,1982].

Example:

The valueset pertaining to street address can be formulated as follows:

```
STRTADDR [street#: (1..1000);
 streetname: string(1..30) of char;
 identifier street#,streetname]
```

The commands for valueset definition can be derived from the user definition commands by substituting 'VALUESET' for 'USER'. Once the valueset type has been defined it can be referenced in the definition of attributes.

ENTITY/RELTIONSHIP/ATTRIBUTE DEFINER

This option provides facilities to define entities, relationships and attributes within an enterprise. The commands supported by this option are:

Entity Definition:

- o ADD ENTITY (entityname)
- o DELETE ENTITY (entityname)
- o CHANGE ENTITY (entityname)
- o LIST ENTITY (entityname)
- o SPLIT ENTITY (entityname)
  as ENTITIES (entitynamel,entityname2)
- o MERGE ENTITIES (entitynamel,entityname2)
  as ENTITY (entityname3)

o RENAME ENTITY (entityname1) as ENTITY (entityname2)

o COUNT ENTITIES

o DEMOTE ENTITY (entityname1)
as ATTRIBUTE (attributename) of ENTITY (entityname2)

Relationship Definition:

- see note below.

Attribute definition:

o ADD ATTRIBUTE (attributename) to ENTITY (entityname)

o DELETE ATTRIBUTE (attributename) of ENTITY (entityname)

o CHANGE ATTRIBUTE (attributename) of ENTITY (entityname)

o MOVE ATTRIBUTE (attributename) from ENTITY (entityname1)
to ENTITY (entityname2)

o LIST ATTRIBUTE (attributename) of ENTITY (entityname)

o RENAME ATTRIBUTE (attributename1) of ENTITY (entityname)
as ATTRIBUTE (attributename2)

o COUNT ATTRIBUTES OF ENTITY (entityname)

o PROMOTE ATTRIBUTE (attributename) of ENTITY (entityname1)
as ENTITY (entityname2)

Note:

The commands for relationship definition can be obtained from the entity definition commands by substituting 'RELATIONSHIP' for ENTITY'. Also, the commands for relationship attribute definition can be obtained from entity attribute definition commands by substituting

'RELATIONSHIP' for 'ENTITY'.

The four commands restricted to entity, relationship and attribute definitions are SPLIT, MERGE, PROMOTE and DEMOTE.

Split Command:

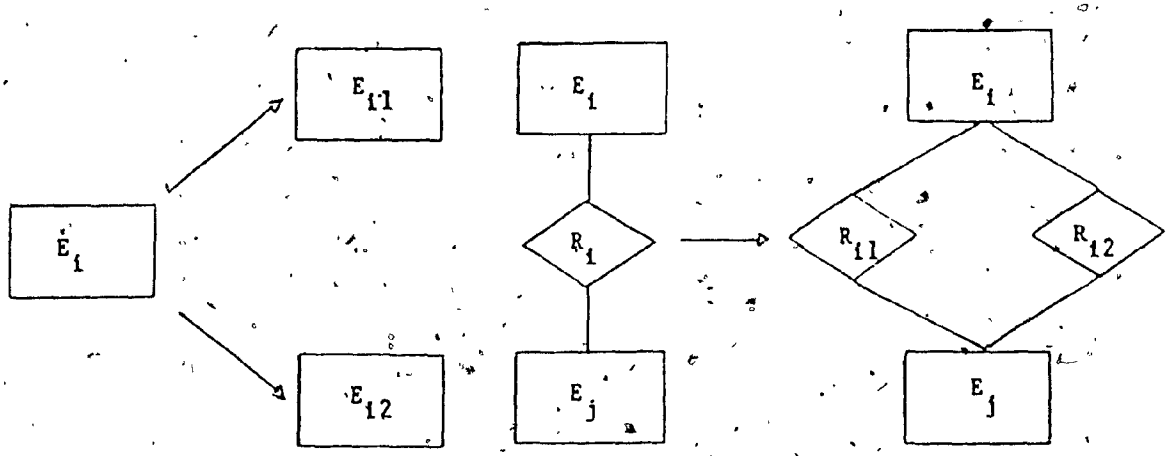
At times, it becomes necessary to split an entity. The split may be necessitated by a new role of the entity in the enterprise, a requirement imposed by the enterprise schema tuning process, etc. The case for relationship splitting is similar to entity splitting. Figure 4.3 describes entity and relationship splits diagramatically.

Merge Command:

A merge achieves the opposite of a split. Again, merge of an entity or relationship can be necessitated by the changing entity or relationship roles within the enterprise, requirements imposed by the enterprise schema tuning process, etc. Figure 4.4 describes the entity and relationship merges diagramatically.

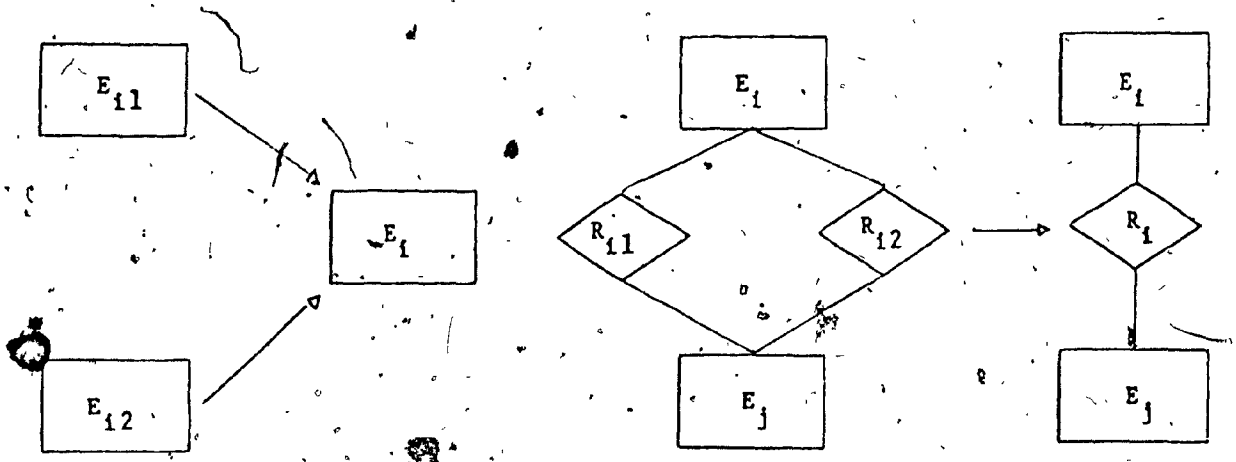
Promote Command:

Due to the evolutionay process within an enterprise, at



LEGEND:
E_i, E_j, E_{i1}, E_{i2} ARE ENTITIES ; R_i, R_j, R_{i1}, R_{i2} ARE RELATIONSHIPS

FIGURE 4.3 ENTITY AND RELATIONSHIP SPLITS



LEGEND:
E_i, E_j, E_{i1}, E_{i2} ARE ENTITIES ; R_i, R_j, R_{i1}, R_{i2} ARE RELATIONSHIPS

FIGURE 4.4 ENTITY AND RELATIONSHIP MERGE

times attributes of an entity (or relationship) can play a role of an entity (or relationship). In otherwords, the candidate attribute can be reckoned as an entity (or relationship) in its own right. The phenomena has been explained by Chen through definition of upper and lower conceptual domains [Chen,1980]. Figure 4.5 describes an attribute PROMOTE diagramatically.

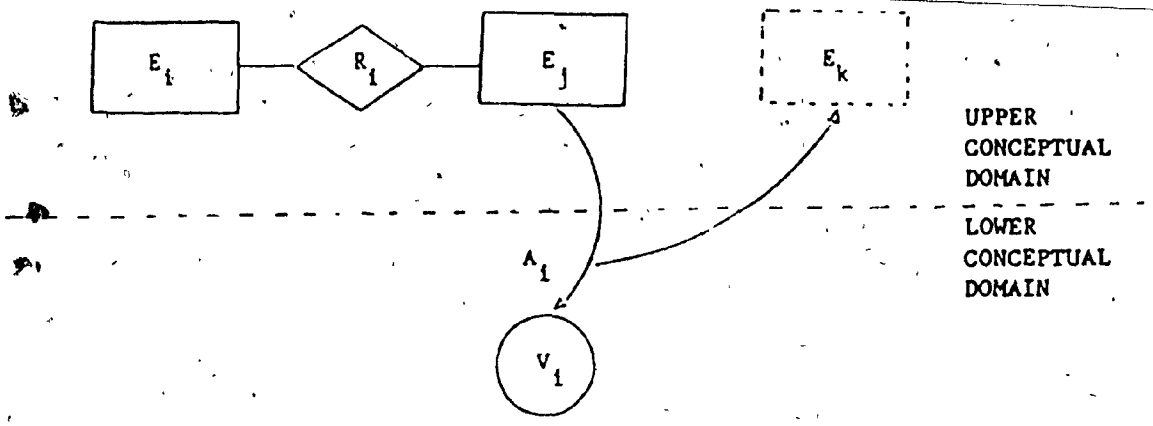
Demote Command:

The demote command achieves the opposite of a promote command. Figure 4.6 describes an entity DEMOTE diagramatically.

4.2.2 CONCEPTUAL SCHEMA GENERATOR

The Conceptual schema generator provides facilities for the formal depiction of conceptual schema in terms of entities and relationships of the enterprise. The conceptual schema generation is menu driven. Upon selection of the conceptual schema generator option, the following menu is displayed:

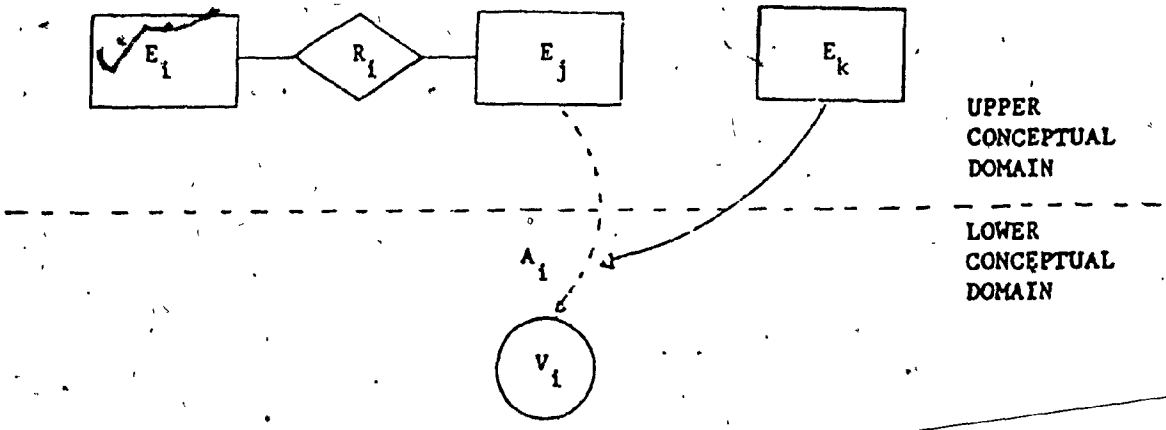
```
CONCEPTUAL SCHEMA GENERATOR:
---------------------------
01. ERD Formatter.
02. ERD Utilities Controller
<3> TYPE-IN SELECTION:
```



LEGEND:

E_i, E_j AND E_k ARE ENTITIES ; R_f IS A RELATIONSHIP ; A_f AN ATTRIBUTE AND V_f A VALUESET

FIGURE 4.5 AN ATTRIBUTE PROMOTE



LEGEND:

E_i, E_j, E_k ARE ENTITIES ; R_f IS A RELATIONSHIP ; A_f AN ATTRIBUTE AND V_f A VALUESET

FIGURE 4.6 AN ENTITY DEMOTE

The conceptual schema generator options are described in the following sections.

ERD (ENTITY-RELTIONSHIP DIAGRAM) FORMATTER

The representation chosen for depicting the conceptual schema is ERD. The ERD Formatter supports creation/update of the E-R diagrams. The facility is supported by the picture oriented data design language interface, which has been discussed in Chapter 3. The facilities offered by the ERD formatter include:

- o Addition, deletion, change, shift and display of entity blocks, relationship blocks, lines, text, etc., that form an ERD
- o Splitting and/or merging entities/relationships
- o Addition, deletion and display of ERD's
- o Storage and retrieval of ERD's into/from DASS meta-data database
- o Temporary hold/retrieval of ERD's into/from storage areas in core
- o Support functions such as:

  ERASE - to erase screen

  GRID - to display a grid for ease of drawing ERD's

  SCALE - to draw horizontal and vertical scales

  LABEL - to label the blocks

  FRAME - to frame EVD's (see Chapter 3, Section 3.4)

The ERD diagram formatting is aided by the graphics screen windowing capability. Two windows are provided where ERD's can be displayed. The ERD's in the window(s) can be used as reference to aid drawing or updating an ERD on the main screen area.. The format of the screen is described in Chapter 5. The valid commands under ERD formatting have been discussed in Chapter 3, under data design language command set. The formated ERD's can be stored in the meta-data database as ERD segments using the PUT command. They can be retrieved from the meta-data database for display or update purposes using the GET command.

ERD UTILITIES CONTROLLER

The ERD utility controller provides access to ERD utilities via the following menu:

```
ERD UTILITIES CONTOLLER:
-----------------------
01. ERD Add
02. ERD Change
03. ERD List
04. ERD Delete
05. ERD Load
06. ERD Chart*
<4> TYPE-IN INPUT:
```

The utilities support the following functions:

- o direct CREATE, UPDATE, LIST, DELETE of ERD.display files
- o loading of off-line coded ERD's
- o creation of ERD chart display files bound for graphics plotter output

Note:

An ERD chart can be thought of as a matrix C, each element, Cij, of which represents an ERD segment. The ERD segments and their order within a chart are defined by the user in conversational mode. Once the ERD chart display file is created by the ERD chart utility, it can be routed to the plotter interface for chart output.

4.2.3 CONCEPTUAL SCHEMA VIEWER

The conceptual schema design is an iterative process. Several iterations of refinement are necessary before the enterprise conceptual model could be generated. Thus during the model building phase, it often becomes necessary to view the enterprise with respect to the partial model. The reasons for this exercise are:

- o to check the validity of the partial model, (i.e., that what has been built fits reality)
- o to discover deficiencies in the partial model, and
- o to identify enhancements required to the partial model

The conceptual schema viewer facility provided by DASS has been designed with the above requirements in mind.

The conceptual schema viewer is menu driven - upon selection of the conceptual schema viewer option, the following menu is displayed:

CONCEPTUAL SCHEMA VIEWER:

---

01. Entity View Displayer
02. Relationship View Displayer
03. Attribute View Displayer
04. Entity-Relationship Structure Displayer
05. Schema (ERD) Displayer

<4> TYPE-IN SELECTION:

The schema views supported by the conceptual Schema viewer are:

- o Entity View - a graphical cum textual display of selected entity
- o Relationship View - a graphical cum textual display of selected relationship
- o Attribute View - a graphical cum textual display of selected attribute of a given entity or relationship
- o Entity-Relationship Structure View - a data structure diagram for selected relationship. The structure diagram consists of entity records connected by single or double headed

uni/bi-directional arrow(s) depending on the relationship type (i.e., 1:1, 1:M, M:1, M:M).

- o Schema (ERD) View - a graphical display of the conceptual schema

The conceptual schema viewer sample outputs are provided in Appendix 4.

4.2.4 CONCEPTUAL SCHEMA VALIDATOR*

This option provides facilities for the validation of the conceptual schema against the information gathered during the conceptual schema design phase. The conceptual schema validation is a responsibility of the data administrator. Once validated, the conceptual schema becomes an input to the database design phase. Here, it is ultimately mapped to the internal (storage) schema and the external (user) schema - see conceptual-internal schema mapper and conceptual-external schema mapper description in Chapter 3.

CHAPTER 5

DASS - INTERNAL DESIGN

5.1 DASS STRUCTURE AND PROCESSING

5.1.1 DASS STRUCTURE

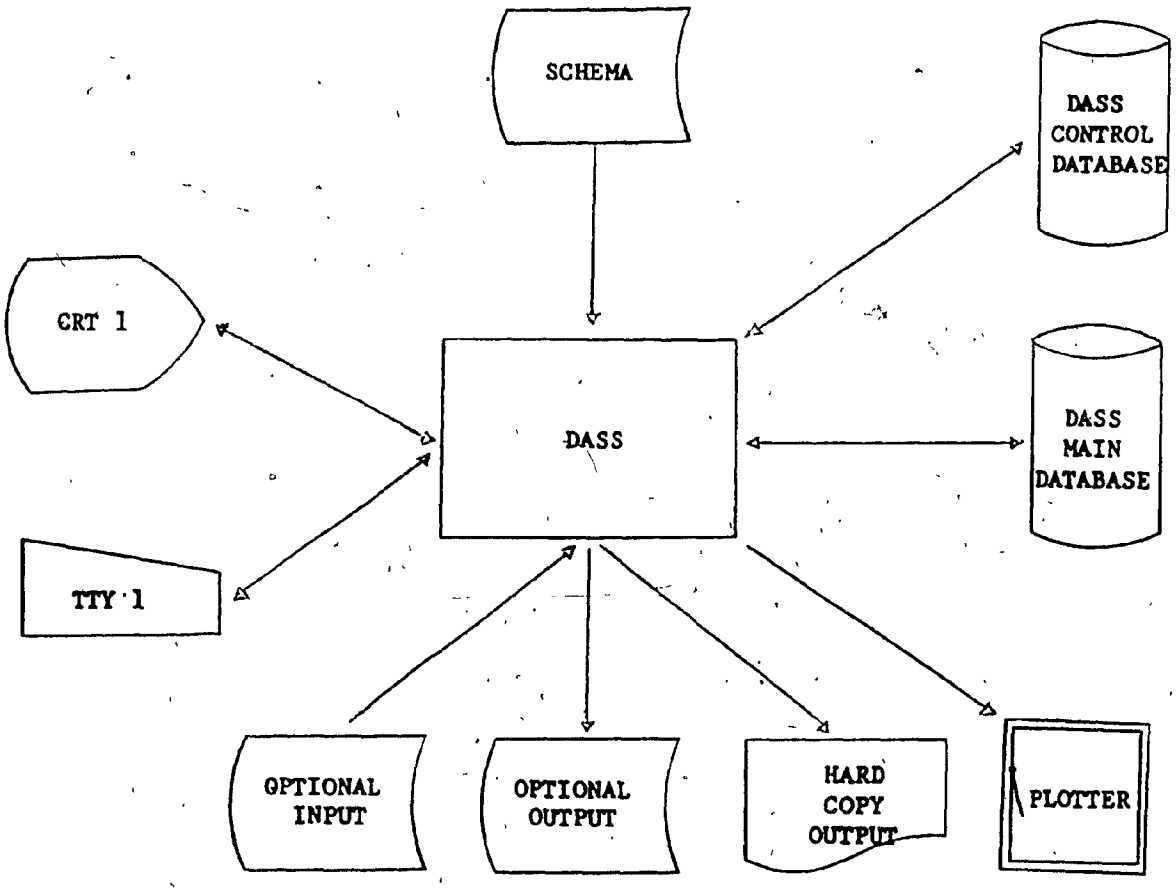
DASS has a modular internal structure, which consists of a hierarcy of modules. The couplings between these modules are those depicted by the hierarchic structure. Thus, only a 'parent' module could invoke its 'child' module. An exception to this rule is the invocation of an 'utility module' - utility modules are 'standalone', could be invoked by any module in the hierarcy, and always return control to the point of invocation. In the following discription the modular structure of DASS is exemplified by:

- o a system context diagram
- o a system Structure diagram which identifies the DASS functional modules, and
- o a set of refinement diagrams for one representative functional module, viz., conceptual schema designer, which describes its internal structure

The diagrams in order are: system context diagram - Figure 5.1, system structure diagram - Figure 5.2, and refinement diagrams - Figures 5.3, 5.4, 5.5 and 5.6.

SYSTEM CONTEXT DIAGRAM

The following diagram describes the context of DASS.



LEGEND:

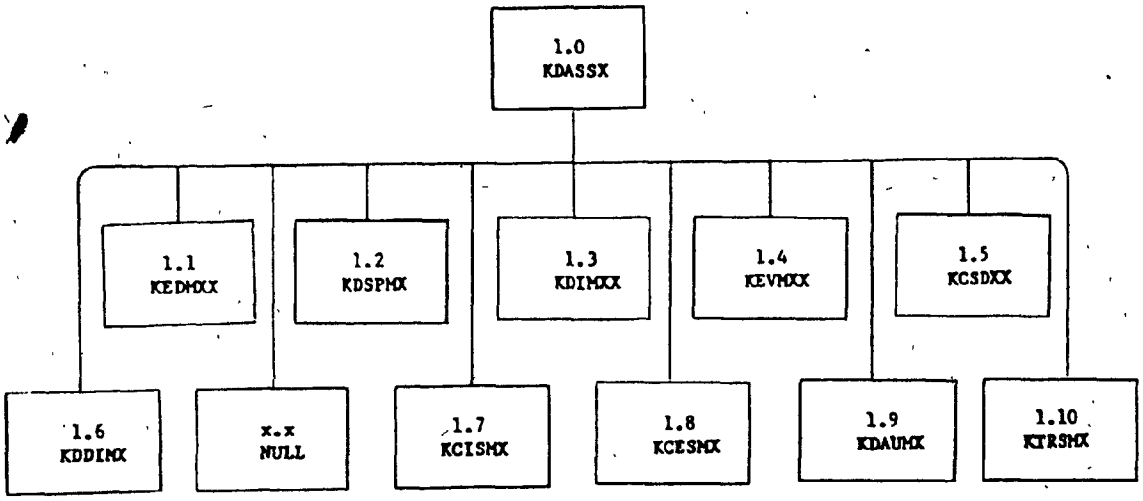
DASS	-	DATA ADMINISTRATION SUPPORT SYSTEM
SCHEMA	-	SCHEMAS FOR DASS MAIN & CONTROL DATABASES
PLOTTER	-	GRAPHICS PLOTTER
CRT 1	-	USER TERMINAL (GRAPHICS TERMINAL)
TTY 1	-	USER TERMINAL (KEYBOARD INPUT/OUTPUT DEVICE)

FIGURE 5.1 DASS SYSTEM CONTEXT DIAGRAM

The hardware/software environment is explained in Appendix 7.

SYSTEM STRUCTURE DIAGRAM

The following diagram describes the structure of DASS from a design/ implementation point of view.



LEGEND:

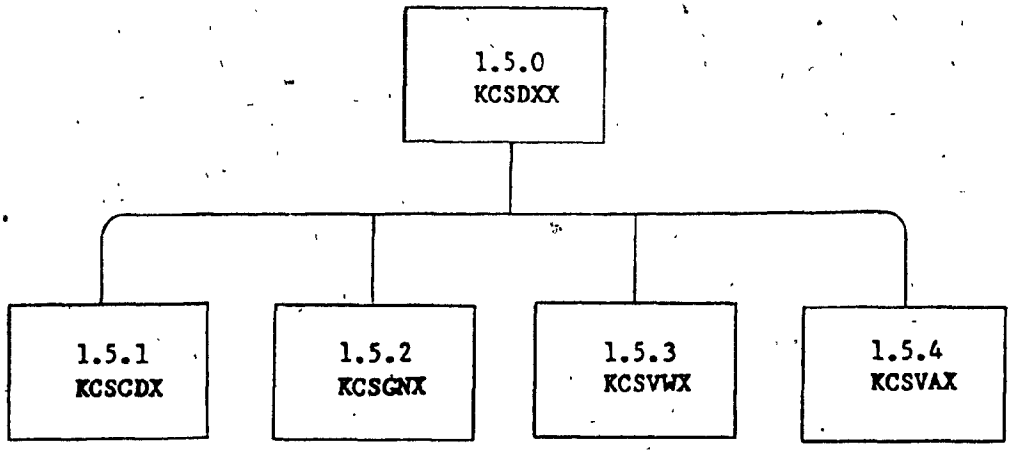
KDASSX - DASS Driver Module
KEDMXX - Enterprise Data Manager Module
KDSPMX - Data Security/Privacy Manager Module
KDIMXX - Data Integrity Manager Module
KEVMXX - Enterprise View Manager Module
KCSDXX - Conceptual Schema Designer Module
KDDIMX - Data Dictionary Interface Manager Module
KCISMX - Conceptual-Internal Schema Mapper Module
KCESMX - Conceptual-External Schema Mapper Module
KDAUMX - Data Administration Utilities Manager Module
KTRSMX - Training & Support Manager Module

FIGURE 5.2 DASS SYSTEM STRUCTURE

There exists a one-to-one correspondence between the modules (DASS driver excepted) listed above and the items on the DASS primary menu.

REFINEMENT DIAGRAM : MODULE 1.5

The following diagram provides a refinement of module 1.5 i.e., Conceptual Schema Designer.



LEGEND:

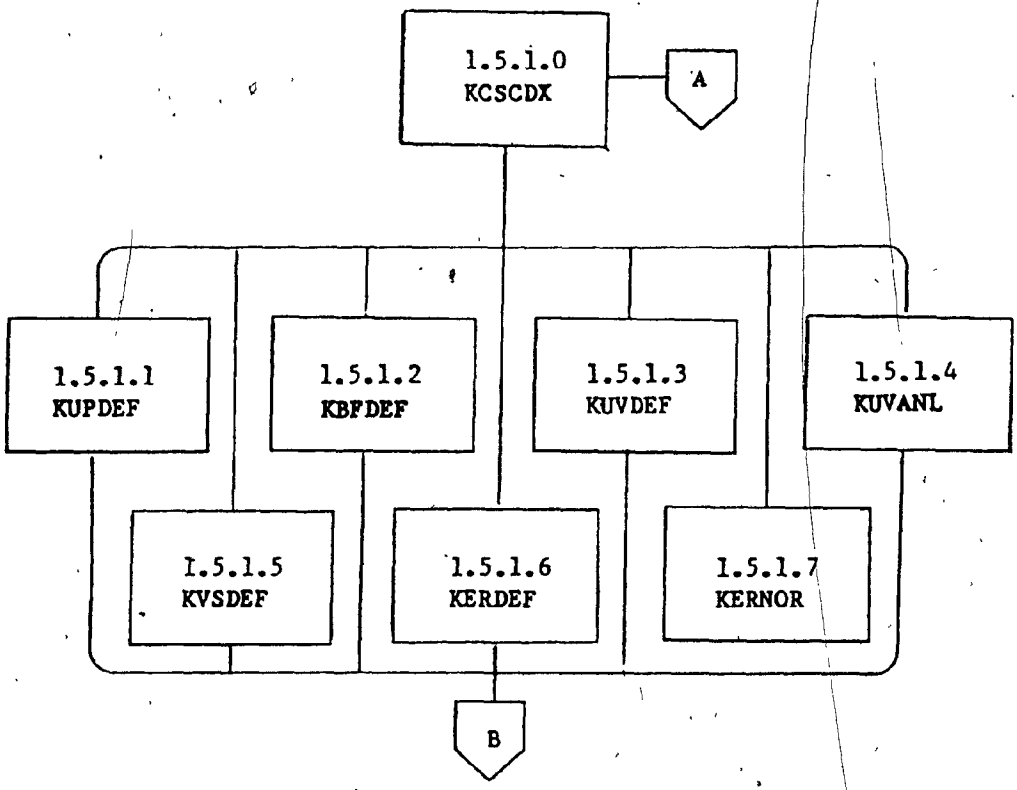
KCSDXX - Conceptual Schema Designer (Driver)
KCSCDX - Conceptual Schema Component Definer
KCSGNX - Conceptual Schema Generator
KCSVWX - Conceptual Schema Viewer
KCSVAX - Conceptual Schema Validator

FIGURE 5.3 CONCEPTUAL SCHEMA DESIGNER

In the current DASS implementation, conceptual schema validation processing is inhibited .

REFINEMENT DIAGRAM : MODULE 1.5.1

The following diagram provides a refinement of module 1.5.1 (i.e., Conceptual Schema Component Definer).



LEGEND:

KCSCDX - Conceptual Schema Component Definer (Driver)

KUPDEF - User Profile Definer Module

KBFDEF - Business Function Definer Module

KUVDEF - Userview Definer Module

KUVANL - Userview Analyser/Synthesiser Module

KVSDEF - Valueset Definer Module

KERDEF - Entity/Relationship/Attribute Definer Module

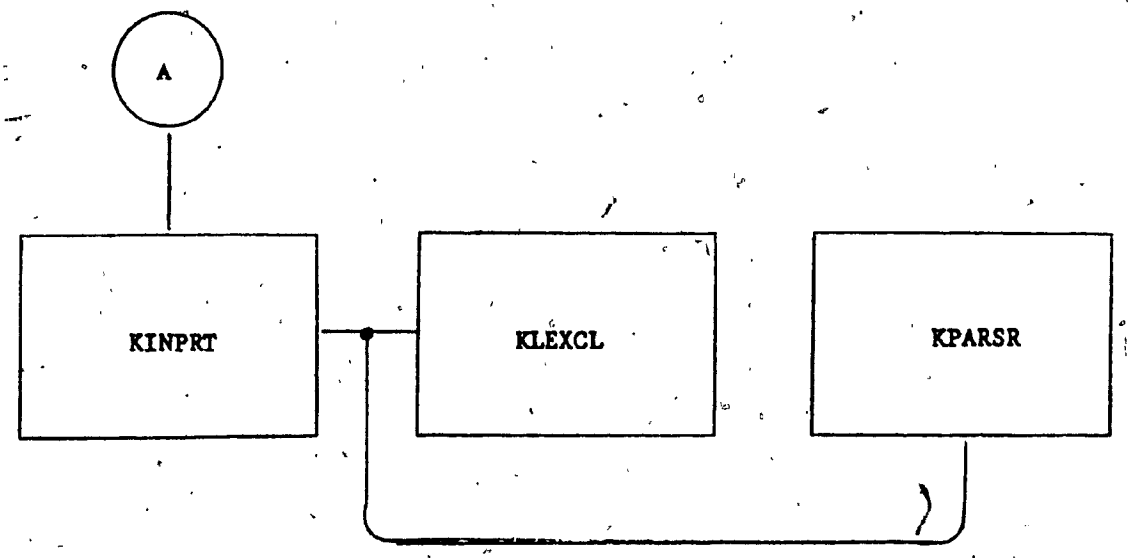
KERNOR - Entity/Relationship Record Normalizer

FIGURE 5.4 CONCEPTUAL SCHEMA COMPONENT DEFINER

In the current implementation, userview analysis/synthesis and entity/relationship record normalization processing is inhibited.

REFINEMENT DIAGRAM : UTILITY 'A'

The following diagram provides a refinement of utility 'A', i.e. Command Interpreter.



LEGEND:

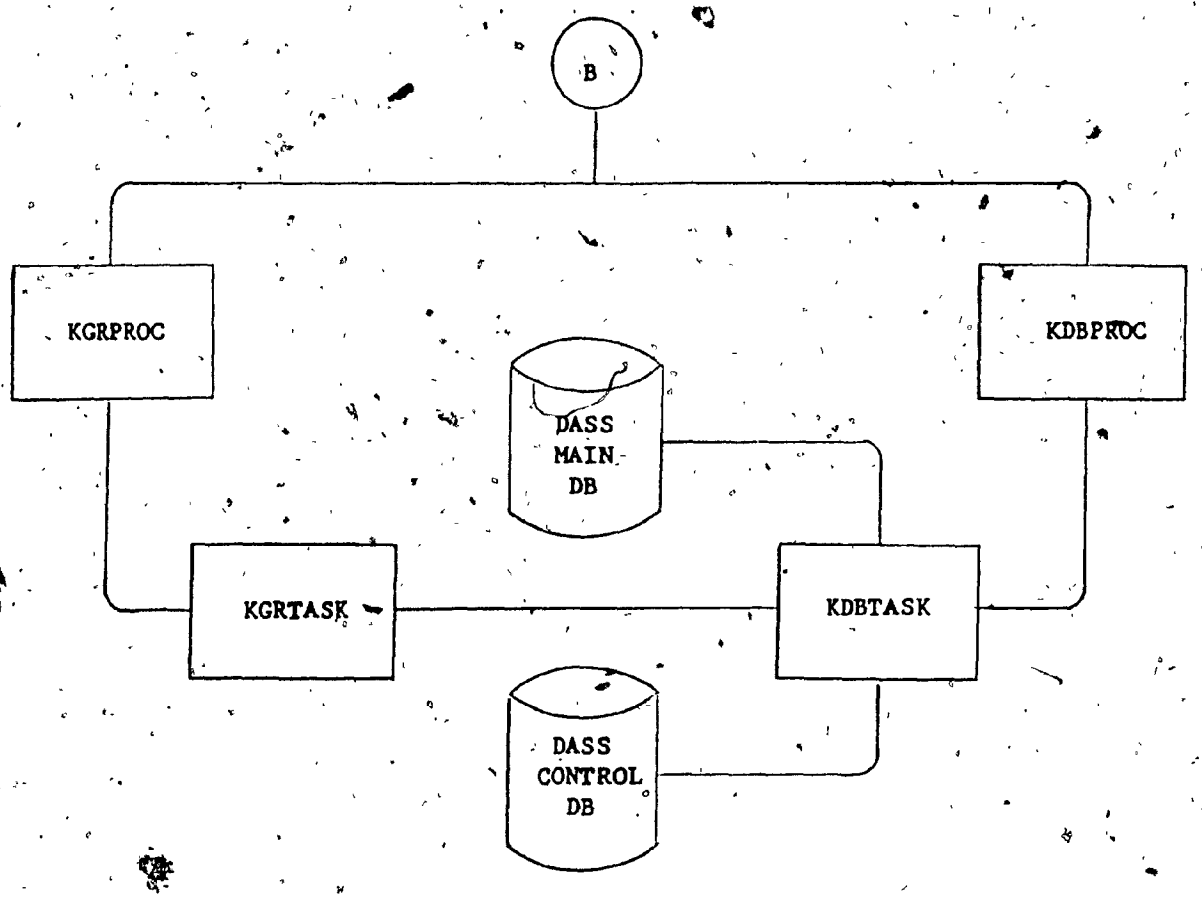
KINPRT - Command Interpreter (Driver)
KLEXCL - Lexical Analyser
KPARSR - Parser

FIGURE 5.5 UTILITY 'A' : COMMAND INTERPRETER

The utility 'A' is used for interpretation of DASS text/picture oriented data design language commands.

REFINEMENT DIAGRAM : UTILITY 'B'

The following diagram provides a refinement of utility 'B' (i.e. Graphics/Database Access Controller).



LEGEND:

KGRPROC - Graphics Facility Access Module (Driver)
KGRTASK - Graphics Task Handler Module
KDBPROC - Database Access Module (Driver)
KDBTASK - Database Task Handler Module

FIGURE 9.6 UTILITY 'B' : GRAPHICS/DATABASE ACCESS CONTROLLER

The database binding occurs in KDBTASK module. Also, graphics facility binding occurs in KGRTASK.

5.1.2 DASS PROCESSING SCHEME

DASS modules, with the exception of utility modules, are invoked in a top-down manner, depending on the function to be completed. The flow of control between modules is always along the hierarchical path. The utility modules are invocable from any module in the main hierarchy, but the converse is not true. The database interface is confined to a single processing module; the same is true for graphic facility interface as well. The prime objective here is DBMS and/or graphics facility replacement flexibility. The single module confinement of database/graphics interface is achieved through postponement of 'process to database' or 'process to graphics' binding time.

5.2 DASS DATABASES - DESIGN, IMPLEMENTATION AND USAGE

The complexity of the data administration task is such that it needs an information system and a database of its own to carry meta-data, i.e., information about information and data [Sundgren,1978]. As described in Figure 5.1, DASS interfaces with two internal databases:

- o DASS main database (meta-data database)
- o DASS control database

The main database is a repository of meta-data in its broad sense, i.e., enterprise descriptions, valueset definitions, security/privacy definitions, integrity-constraint

definitions, entity descriptions, entity-relationship descriptions, attribute descriptions, etc.

The control database is a repository of data required for the proper working of DASS. The data includes parser tables, command/command-synonym tables, graphics screen/symbol definition tables, dialogue control tables, error diagnostics tables, and tutorial module tables.

5.2.1 DASS MAIN DATABASE

LOGICAL DATABASE STRUCTURE

The main data domain consists of 23 entitiies, and 22 relationships - see record and set types described below. The logical database model for the main database is described by the entity-relationship diagram of Figure 5.7.

DATABASE IMPLEMENTATION

The logical data structure is implemented using the network data model. The DBMS used is GPLAN - Generalized Planning System [Haseman,1976; Haseman,1977]. The entities and relationships shown in the logical structure diagram are implemented as 'record types' and 'set types' of the network model, respectively.

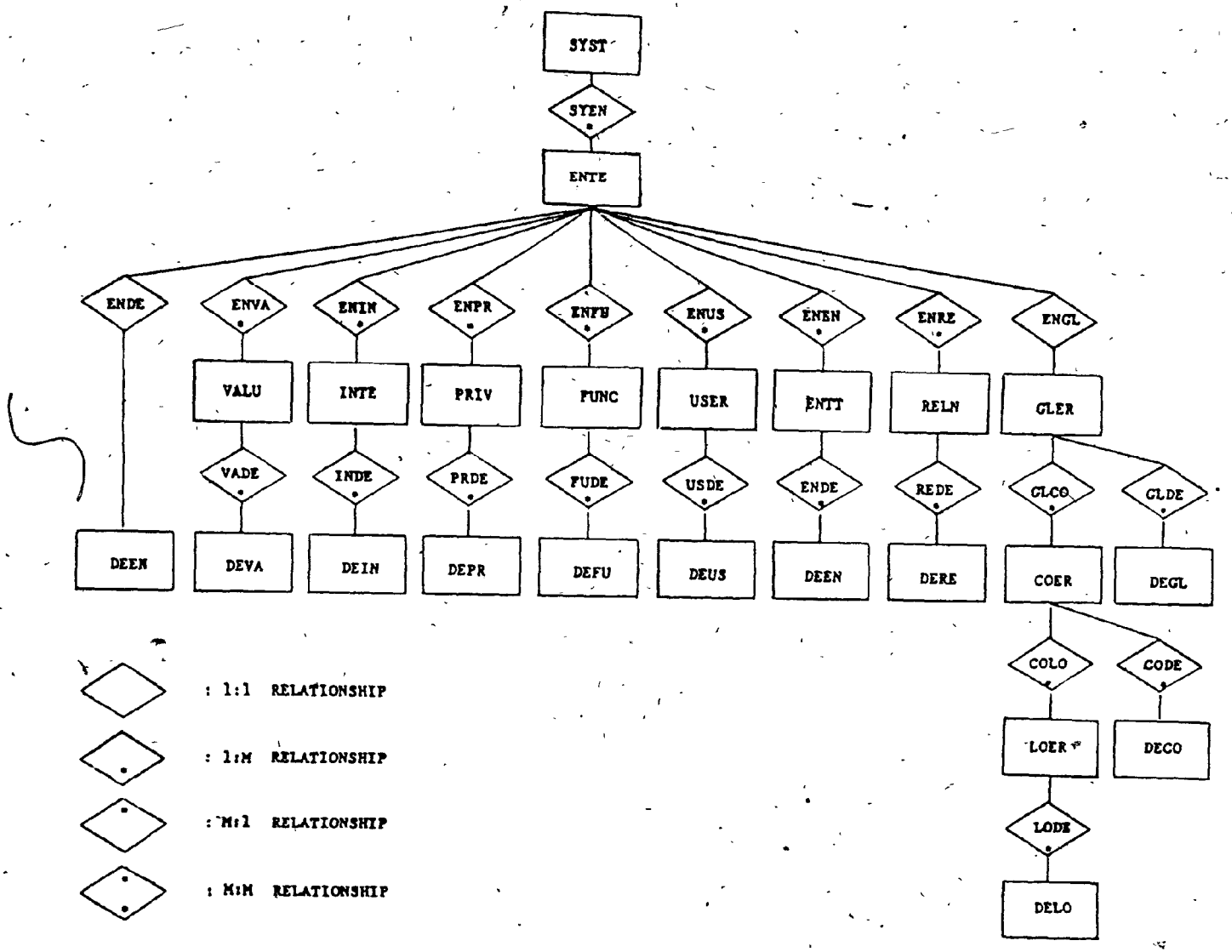


FIGURE 5.7 DASS META-DATA DATABASE - LOGICAL DATA MODEL

Database Record Types:

Type	Explanation
SYST	GPLAN internal record
ENTE	Enterprise definition
VALU	Valueset definition
INTE	Integrity-constraint definition
PRIV	Privacy/security definition
FUNC	Function definition
USER	User profile definition
ENTT	Entity definition
RELN	Relatiionship definition
GLER	Global view definition
COER	Contextual view definition
LOER	Local view definition
DEEP	Enterprise description
DEVA	Valueset description
DEIN	Integrity-constraint description
DEPR	Privacy/security description
DEFU	Function/Process/Activity description
DEUS	User/Userview description
DEEN	Entity/Attributes description
DERE	Relationship/Attributes description
DEGL	Globalview description
DECO	Contextualview description
DELO	Localview description

Database Set Types:

Type	Explanation
SYEN	System-Enterprise relationship
ENVA	Enterprise-Valueset relationship
ENIN	Enterprise-Integrity relationship
ENPR	Enterprise-Privacy relationship
ENFU	Enterprise-Function relationship
ENUS	Enterprise-User relationship
ENEN	Enterprise-Entity relationship
ENRE	Enterprise-Relationship relationship
ENGL	Enterprise-Globalview relationship
GLCO	Global-Contextualview relationship
COLO	Contextual-Localview relationship
EPDE	Enterprise-Description relationship
VADE	Valueset-Description relationship
INDE	Integrity-Description relationship
PRDE	Privacy-Description relationship
FUDE	Function-Description relationship
USDE	User-Description relationship
ENDE	Entity-Description relationship
REDE	Relationship-Description relationship
GLDE	Globalview-Description relationship
CODE	Contextualview-Description relationship
LODE	Localview-Description relationship

DATABASE USAGE

The DASS user commands that require access to the main database are routed to the 'KDBPROC' module. Here, the type of access required is analysed, I/O table is built and then control is passed to the 'KDBTASK' module. The actual database access is carried out by 'KDBTASK' via one or more data manipulation language (DML) commands. These commands are assembled and invoked according to the information provided in the database I/O table.

5.2.2 DASS CONTROL DATABASE

LOGICAL DATABASE STRUCTURE

The control data domain consists of seven entities - see record types described below. Since these entities are functionaly independent of each other, there exist no defined relationships between them. The logical data model for the control database is described by the entity-relationship diagram of Figure 5.8.

DATABASE IMPLEMENTATION

The logical data structue is implemented as a flat file with sequential access organization. Each entity in the logical data structure is implemented as a record type:

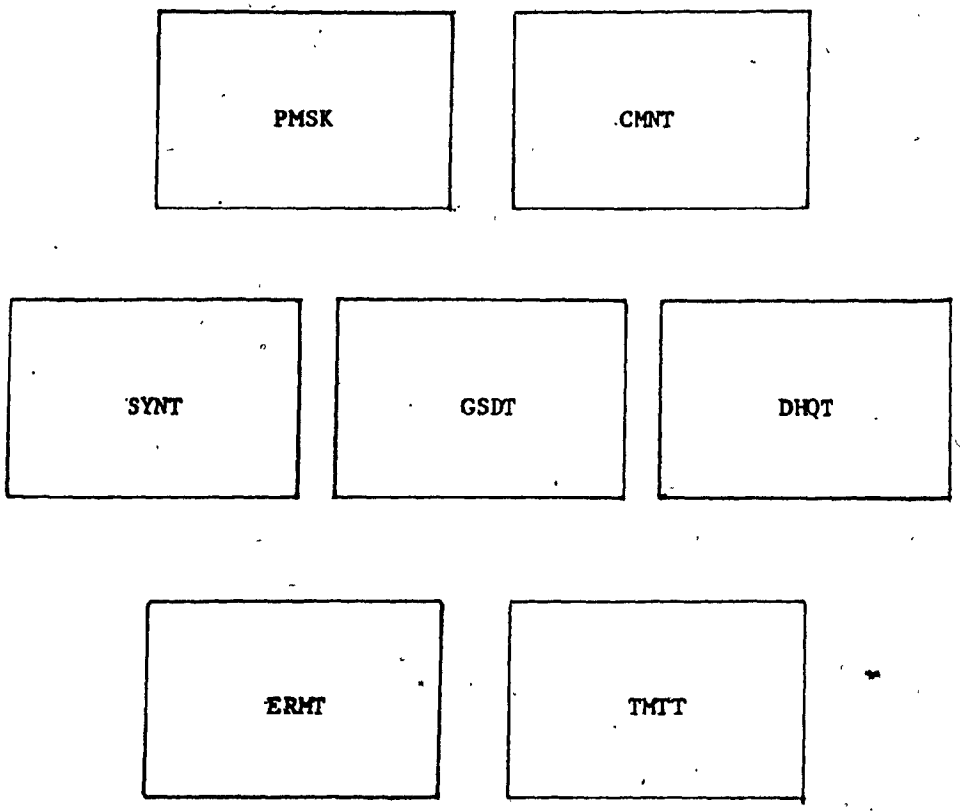


FIGURE 5.8 CONTROL DATABASE - LOGICAL DATA MODEL

Database Record Types:

Type	Explanation
PMSK	Parser mask table
CMNT	Command mnemonic table
SYNT	Command synonyms table
GSDT	Graphic screen/symbol definition table
DHQT	Dialogue handler query profile table
ERMT	Error diagnostics/messages table
TMTT	Tutorial modules text table

DATABASE USAGE

The data contained in the DASS control database is used by DASS processes. At DASS invocation time, the parser, command mnemonic, synonym, diagnostics/messages and graphic screen/symbol definition tables are automatically initialized by loading the corresponding data from the control database. The other tables are loaded on a need basis. (Note: It is possible to alter data on the control database and then re-initialize any desired tables - this feature permits customization of DASS processing).

5.3 DASS GRAPHICS FACILITY DESIGN

The graphics facility is intended to provide capabilities for ERD and EVD drawing/manipulation:

- o create, update, delete and store ERD's and/or EVD's
- o retrieve ERD's and/or EVD's
- o temporary 'HOLD' of ERD's and/or EVD's in core for rapid recalls, and
- o formatting of ERD and/or EVD charts

5.3.1 DASS GRAPHICS SCREEN FORMAT

The graphic facility used by the current implementation is a Tektronix 4010 terminal (Tektronix,1979) with a screen copier and a graphics plotter (for details on the hardware/software environemnt, see Appendix 7). The graphics screen consists of 1024 x 780 addressable points - the screen partitioning scheme for DASS processing is shown in Figure 5.9.

The window1 and window2 areas are used to display existing ERD's or EVD's for reference purposes - the ERD/EVD drawing and editing is always done on the main screen. The display on windows is achieved by drawing the desired ERD or EVD on the virtual screen and projecting it onto the main screen. The main screen layout for ERD/EVD drawing is shown in Figure 5.10. As it can be seen, the screen layout resembles a GRID with six rows (numbered 1 thru 6) and six columns (numbered 1 thru 6). Each slot generated by the GRID is assigned a two-digit label. The label Sij for the slot on the Ith row and Jth column is given by the formula:

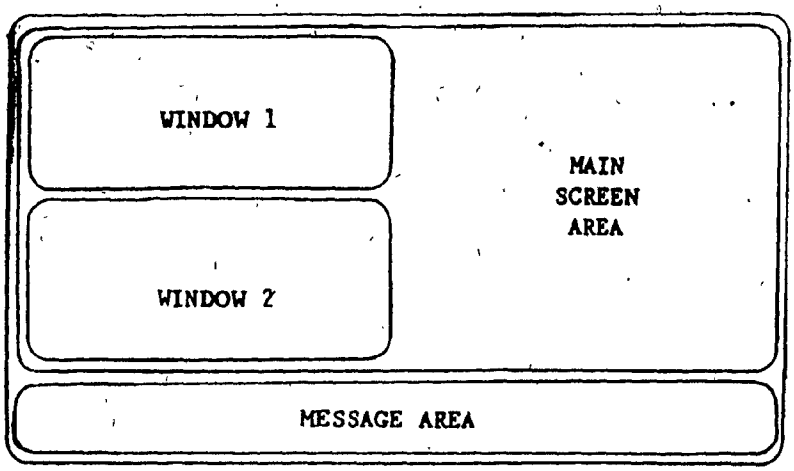


FIGURE 5.9 GRAPHICS SCREEN LAYOUT

	1	2	3	4	5	6
1	.11	.12	.13	.14	.15	.16
2	.21	.22	.23	.24	.25	.26
3	.31	.32	.33	.34	.35	.36
4	.41	.42	.43	.44	.45	.46
5	.51	.52	.53	.54	.55	.56
6	.61	.62	.63	.64	.65	.66

FIGURE 5.10 DASS MAIN SCREEN LAYOUT*

* Default layout under the current implementation

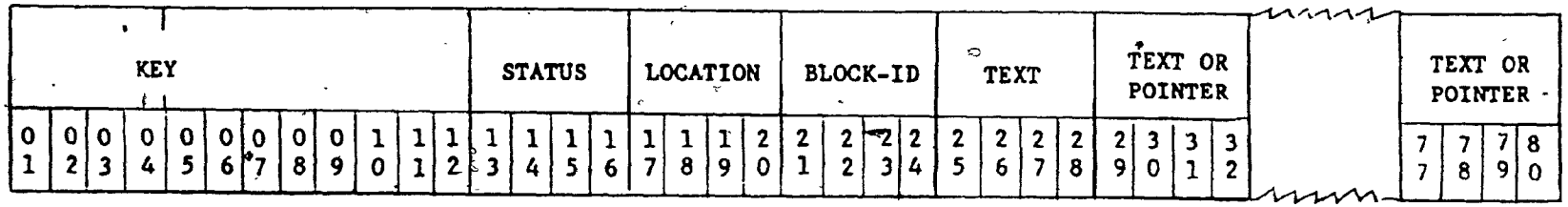
Sij = 10i + j.

The slots on rows 1 and 6 are used for text strings - text strings can be placed across a slot or across an entire row. The slots on rows 2 through 5 can contain both picture and textual data. The center coordinates for the slots are pre-defined and stored in the graphics screen definition table. To draw an ERD/EVD symbol at some slot Sij, a table look-up is initiated (to extract the center coordinate for the slot Sij) and control is passed to the desired symbol drawing procedure. All ERD/EVD symbols are drawn using the relative co-ordinates defined for the symbol. The symbol set used by the current implementation includes symbols for entity, relationship and variations of these. The symbol set can be extended or existing symbols can be altered by applying the updates to the control database.

5.3.2 DASS GRAPHICS DISPLAY FILE

A graphics display file provides a data storage scheme for ERD/EVD display information. The DASS display file record format is shown in Figure 5.11.

The simple format of the display file facilitates reading it like a report. Also, the convenient record format/length facilitates rapid off-line coding/creation and update of display files. These display files, with display records



LEGEND:

KEY - ERD/EVD Identification + Sequence Number

STATUS - Symbol Status : * = Deleted, □ = Active

LOCATION - Screen Slot Number

BLOCK-ID - Symbol Identification (ID)

TEXT - Text to be displayed in the Slot

POINTER - Connectivity (to other symbol) Identifier

FIGURE 5.11 DASS DISPLAY FILE FORMAT

delimited by control cards ($DATA <datatag> and $END) can be directly read into the database by the graphics utility provided by the conceptual schema viewer option of DASS. Powerful data structures currently exist in interactive graphics systems to provide performance-effective design/drafting facilities. Intricate data structures, however, are difficult to program, debug, maintain, and modify [Cardenas,1975; Newman,1979]. As it can be seen, the graphics data structure of DASS provides a simple scheme for representation of display data of two dimensional objects.

5.3.3 DASS GRAPHICS ACCESS

The Tektronix software consists of a number of FORTRAN subroutines invocable from user programs for accomplishing a variety of graphics tasks. All user commands that require graphics access for their execution are routed to 'KGRPROC' module. In 'KGRPROC' module, the type of graphic access required is analysed, I/O table is built, and then control is passed to 'KGRTASK' module. The actual graphic access is carried out by 'KGRTASK', which invokes one or more graphics subroutines, according to the information provided in the graphics I/O table.

PICTURE DISPLAY AND EDITING

The picture oriented data design language component provides commands for ERD/EVD creation and editing. Once

created, ERD/EVD's are stored in the main database using the 'PUT' command and are retrieved using the 'GET' command. The 'GET' command retrieves the display file from the database and places it in the display buffer. Once the ERD/EVD information is in the buffer, it can be displayed (on the main screen, window1 or window2), edited, erased or moved to one of the 'HOLD' areas. The commands like 'PUT ERD into HOLD #n' and 'GET ERD from HOLD #n' can be used to transfer/retrieve current ERD to/from 'HOLD' areas. The HOLD capability facilitates rapid browsing of several ERD/EVD's, in succession, without having to access the database.

GRAPHICS PLOTTER FACILITY*

The 'view utility manager' of DASS provides the capability to define ERD and EVD charts. An ERD/EVD chart, as described in the previous Chapter, can be thought of as a matrix C, where each matrix element, Cij, represents an ERD/EVD segment. The view utilities manager lets the user define each Cij, interactively. Once the matrix has been defined, the utility retrieves display data for the defined ERD/EVD segments from the database and creates a display file. This display file can then be passed to the plotter interface to draw the chart. The plotter hardware is described in Appendix 7.

5.4 DASS DIALOGUE HANDLING

The tools used for DASS-user dialogue are MENUS and COMMAND languages. As described earlier, DASS has two command languages: a control command language and a text/picture oriented data design language.

5.4.1 DASS MENU HANDLING

As described in Chapter 3, DASS has one primary menu and several secondary menus. The display of each menu is followed by a 'prompt' to the user to enter a menu selection number, corresponding to the desired processing option. If the user response is a control command, then appropriate action is taken and the user is re-prompted for menu selection.

5.4.2 DASS CONTROL COMMAND HANDLING

The control commands are recognized by the presence of a leading '$' sign in the input string. The control command format is described below:

$<commandverb>,<commandoption> <commandspec>

Only the first three characters in the commandverb are considered significant. The commandoption is an alpha character that indicates which of the several processing

modes is desired, and commandspec is the data required by the control command for its execution.

Example:

```
$HELP,E IN1090
```

This command requests an explanation of the error code 'IN1090'. Here, commandverb is 'HELP', commandoption is 'E', and commandspec is 'IN1090'.

Note:

A complete list of control commands is provided in Chapter 3 (see Section 3.3).

5.4.3 DATA DESIGN LANGUAGE COMMAND HANDLING

The text/picture oriented data design language syntax, in BNF notation, is given below:

```
<command>        ::= <commandverb> <actionlist>
<actionlist>     ::= <whatcomponent>
                     <wherecomponent> <howcomponent>
<commandverb>    ::= ADD/CHANGE/DELETE/MOVE/LIST/
                     SPLIT/MERGE/RENAME/COUNT/PROMOTE/
                     DEMOTE/DISPLAY/GET/PUT/ERASE/
                     GRID/SCALE/LABEL/FRAME
<whatcomponent>  ::= <whatspec1>/
                     <whatspec1>,<whatspec2>/
```

<whatspec1>,<whatspec2>,<whatspec3>/

<whatspec1>,<whatspec2>,<whatspec3>,

<whatspec4>/

The sentential forms for <wherecomponent> and <howcomponent> are similar to those of <whatcomponent>. The lower level sentential forms have been described using free-format, in Chapter 3.

The text/picture oriented data design language command interpretation comprises of four steps:

- o Command Mnemonic Validation
- o Command Normalization
- o Lexical Analysis, and
- o Parsing

These steps are executed in sequence. If errors are encountered while processing, command interpretation is terminated, appropriate error message is displayed and user is prompted to re-type command input. The command intepretation steps are discussed below:

COMMAND MNEMONIC VALIDATION

The processing in this step consists of validation of the commandverb portion of the input string. The validation logic is explained by the algorithm shown in Figure 5.12.

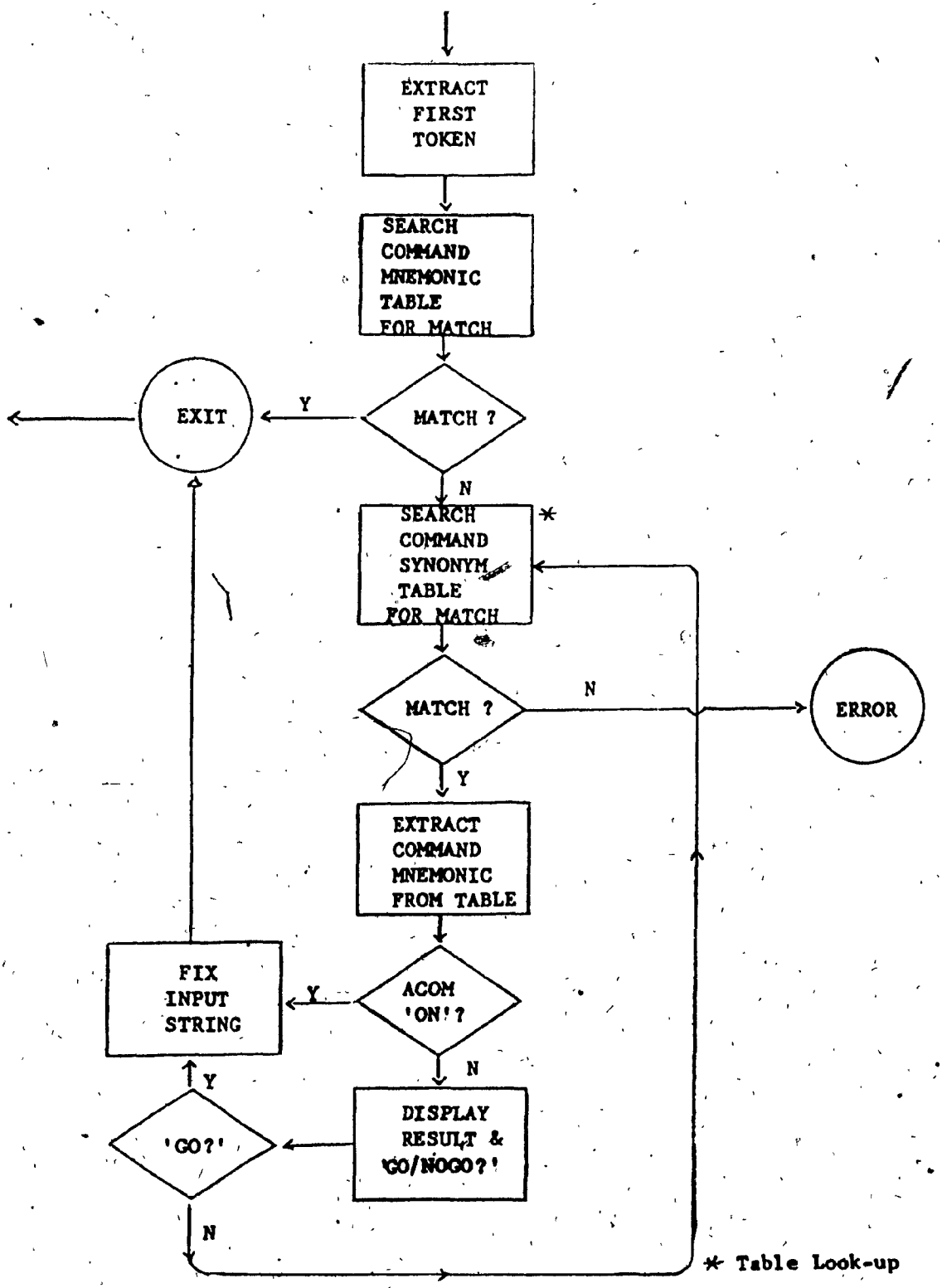


FIGURE 5.12 COMMAND MNEMONIC VALIDATION LOGIC

Upon completion of commandverb validation, the input string is passed to the command normalization step.

COMMAND NORMALIZATION

In order to provide an English-like resemblance, the data language syntax tolerates several words which from a command semantics point of view are superfluous. Also, the language accepts synonyms in place of certain keywords. It is therefore necessary to normalize the input string prior to interpretation. This is done by noise-word deletion and synonym token replacement.

Noise Word Deletion: This process deletes from the command input string such noise words as 'a', 'the', 'an', 'for', 'of', etc., which are superfluous.

Synonym Token Replacement: This process searches for tokens 'to', 'and', '&', 'between', and replaces them by a comma. Also, certain synonym strings are searched and replaced by their equivalent keywords; example: 'integrity-constraint' is replaced by 'integ' keyword.

The normalized string from this step is passed to the lexical analysis step.

LEXICAL ANALYSIS

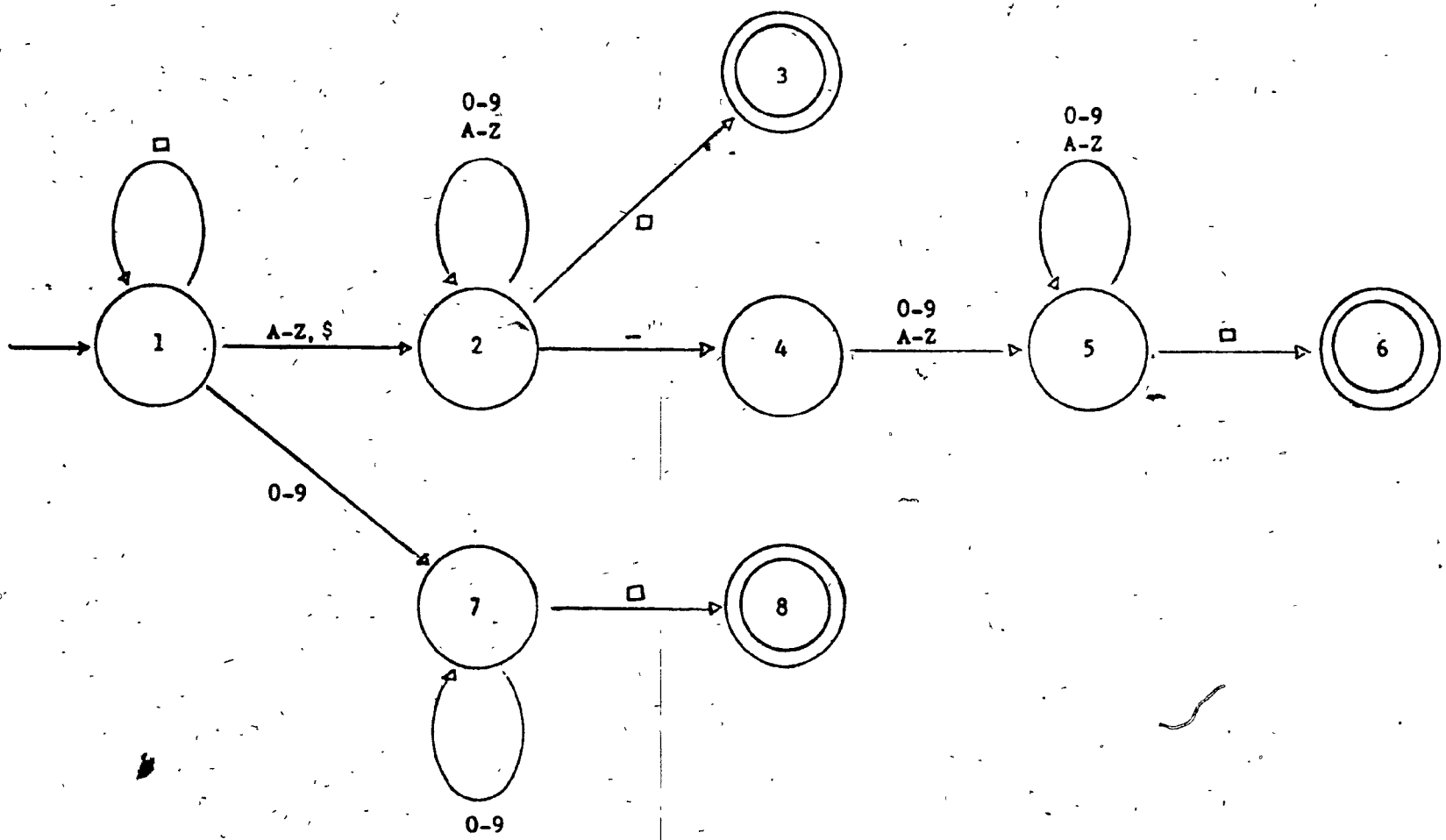
The objective of the lexical analysis is to scan the command string, recognize the tokens one by one, and load these tokens into the symbol table. The finite state automata used by the lexical analyser is described in Figure 5.13.

The symbol table (see Table 5.1) used by the lexical analyser is a 4 x 4 matrix, each row corresponds to tokens associated with the 'commandverb', 'whatcomponent', 'wherecomponent', and 'howcomponent' parts of the command, respectively.

The punctuation marks recognized by the lexical analyser are not stored in the symbol table. However, the fact of their presence or absence is used to determine the symbol table slot where the next non-trivial token is to be stored. Table 5.2 shows the contents of the symbol table after a lexical analysis of the following string:

RENAME ATTRIBUTE(beta) of ENTITY(alpha) as ATTRIBUTE(gamma)

In Table 5.2, the '$' sign in front of a token indicates that it is a non-keyword. A token is validated as being a keyword by a keyword table lookup. Upon completion of the syntax analysis of the input string, the symbol table is



LEGEND:

□ = BLANK OR SPECIAL CHARACTER ('-', AND ' $ ' EXCLUDED)

FIGURE 5.13 LEXICAL ANALYZER - FINITE STATE AUTOMATA

command	-	-	-
whatspec1	whatspec2	whatspec3	whatspec4
wherespec1	wherespec2	wherespec3	wherespec4
howspec1	howspec2	howspec3	howspec4

Table 5.1 DASS SYMBOL TABLE STRUCTURE

RENAME	-	-	-
ATTR	$BETA	ENTT	$ALPHA
ATTR	$GAMMA	-	-
-	-	-	-

Table 5.2 DASS SYMBOL TABLE WITH ENTRIES

passed to the parsing step.

PARSING

The parsing of the tokens consists of building a parser mask (using the information from the symbol table) and comparing it against a set of pre-defined masks. If the 'compare' results in a 'HIT', then the original command is accepted and its execution is initiated. If not, then the syntax error is displayed and the user is prompted to re-enter the command. The parser mask is built as follows:- the symbol table is scanned left to right, top to bottom, one slot at a time. A parser process 'P' is used to generate parser mask component P(Aij) for the content of each slot Aij. The Aij --> P(Aij) mapping is described in Table 5.3.

In the current implementation, P(Aij) for i=1 and j=2,4 are not used in the mask, as Aij values are always blank in this case. Thus, the parser mask generated for the command RENAME ATTRIBUTE(alpha) of ENTITY(beta) as ATTRIBUTE(gamma) is given by the string : 'RAIEIAI******'. All the valid parser masks are stored in a internal mask table, which is loaded from the control database. The parser mask matching process identifies the type of the current command. Once the command type has been identified, appropriate module is invoked for its execution.

Aij	P(Aij)
blank	'*'
keyword	unique keyword-id
non-keyword	'I'

NOTE:

The parser mask is generated using the formula:

M = //P(Aij), i=1,4;j=1,4

Where:

M : is the generated parser mask

// : is the concatenation operator

i,j : indices of the symbol table

Table 5.3 PARSER-MASK MAP DESCRIPTION TABLE

5.5 DASS OPERATING ENVIRONMENT CONTROL

DASS supports customization of system processing. The required operating environment is created by setting 'ON' or 'OFF' the necessary control flags.

5.5.1 DASS PROCESS CONROL FLAGS

ECHO - User input is echoed based on the value of this flag

PLOT - The value of this flag indicates availability of graphics facility

QUIK - The user preference for brief output is indicated by this flag

ACOM - DASS supports use of synonyms in place of actual command verbs. Once the equivalent command verb for a synonym has been identified, DASS has two processing paths, viz., proceed to execute the command or prior to execution prompt the user to check the command verb substitution. The value of ACOM identifies to DASS as to which path should be taken

TEST - If this flag is set, then all updates to the database are considered as 'NULL'. The test flag is checked prior to each database write operation

DBUG - During DASS processing, information captured at key points is stored in internal work areas. By setting DBUG 'on', this information could be displayed. The displayed information provides a step by step history of command execution which is useful for DEBUG purposes

MCOM - If MCOM is set 'on' then DASS displays date/time at receipt of a command and at the end of excution of that command. The information provided by DASS is useful in monitoring the performance of DASS commands, dynamically

MDBA - MDBA is similar to MCOM but here the date/time stamping applies to database access

MGRA - MGRA is similar to MCOM but here the date/time stamping applies to graphics facility access

Note:

The default setting of ACOM is 'ON'; all other flags are set to 'OFF' state.

5.6 DASS ERROR HANDLING

The error and warning messages are handled at the module level. All messages are prefixed by a twelve

character error code. The error code comprises of two parts - an internal code part and an external code part. The error code structure is shown in Table 5.4.

Example:

'**ERROR** ERTR01-IN1090 INVALID INPUT... RE-TRY'

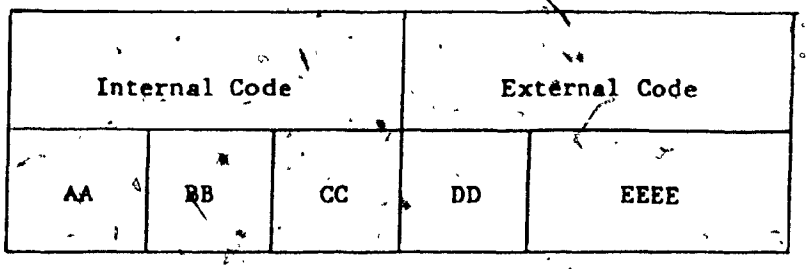
Here, the internal error code is ERTR01 and the external error code is IN1090. The error message entries on the control database have only the external error code as key. Consequently, to obtain explanation the user only need supply the external error code.

5.7 DASS ACCOUNTING

The DASS accounting procedure is responsible for the capture and display of the following information:

- o Start date/time of DASS session
- o CPU usage
- o DASS session exit type
- o Number of database calls
- o Number of graphics calls
- o DASS usage charge
- o Finish date/time of DASS session

One of the objectives of providing the accounting



LEGEND:

AA : Error(ER) or warning(WA) message indicator

BB : A two character mnemonic of the module issuing the message

CC : A local error/warning sequence number that is unique within each module

DD : A two character mnemonic identifying a conceptual error class (eg., INPUT, PARSER, DBACCESS, etc.)

EEEE : A global error/warning sequence number that is unique within DASS

Table 5.4 DASS ERROR CODE STRUCTURE

information is to keep the user aware of the use/abuse of system resources. Dass keeps track of system usage by maintaining a number of counters, one each for CPU usage, DB access, graphics access, etc. The DASS usage charge is computed using an accounting algorithm.

CHAPTER 6

DASS - THE WORK AHEAD

6.1 DASS - FUTURE DIRECTIONS AND CONCLUDING REMARKS

6.1.1 DASS FUTURE DIRECTIONS

DASS in its current form appears to be a valuable tool in the area of data administration. The more readily visible features of DASS are: ease of conceptualization of DASS facilities, friendliness of DASS user interface, and ease of enterprise analysis via stepwise view refinement. DASS, as described earlier, is a collection of sub-systems of which three sub-systems, viz., conceptual schema designer, enterprise view manager, and training & support manager, have been implemented to-date. Therefore, the immediate task ahead is the implementation of the remaining sub-systems. With the overall structure of DASS being already in place, this task is pretty straightforward. Apart from the targetted sub-systems implementation, several other enhancements could be made to DASS. The functional areas for potential enhancements are discussed below.

DATA BASE DESIGN

Several design modifications can be considered for the main database. For instance, new relationships such as

'entity-relationship', 'user-entity', 'function-entity', etc. can be introduced to support diverse needs. Also, the current 'enterprise' record type could be split into 'super-enterprise' and 'sub-enterprise' record types with a 1:M relationship between the two. Such an approach could greatly ease enterprise studies at a sub-enterprise or business-unit level.

DATA LANGUAGE

The command language can be extended to support several new functions. For instance, a 'FIND' or 'LOCATE' command can be defined which can be used as follows:

LOCATE RELATION ? (entity1,entity2)

i.e., find a relation which has entity1 and entity2 as participating entities.

LOCATE RELATION ? (entity1;*)

i.e., find a relation which has entity1 as a participating entity.

Other useful commands in this category are:

FIND { USER / FUNCTION / RELATIONSHIP } THAT USES ENTITY (identl)

LIST { ENTITIES / RELATIONSHIPS } USED BY { USER / FUNCTION } (identl)

etc...

FULL SCREEN CAPABILITY

A full screen capability is a desirable option for DASS. It can greatly enhance the user-machine interface in such areas as menu display, option selection, data entry, editing, etc.

MULTI-TASKING CAPABILITY

When extensive database searches are involved some of the DASS processes might tend to be time-consuming. Invariably, valuable user-time is expended when an online process is waiting for completion of a background task. The multitasking feature can be considered to circumvent this problem.

MULTI-LEVEL DIALOGUES

The software tools in general provide fixed-format system-user dialogue facilities. It is a recognized fact that there are different levels of human interfaces to software products. Therefore, a more realistic approach would be to offer multi-level dialogue features to cater to the needs of a spectrum of users from novice to expert.

PRE-DEFINITION OF INPUT

Considerable time and effort is spent during online sessions on entering input which might remain constant, session to session. An ideal enhancement would be a capability to catalogue procedures that could be invoked to interpose the desired input. With such facility, the user would be able to by-pass some of the time-consuming menu selection steps.

BATCH PROCESSING CAPABILITY

DASS could be modified to support access in batch mode. The batch access could be used for such transactions which are pre-definable.

GRAPHICS FACILITIES

The current DASS software can be enchnced to provide interface to different types of graphics devices: refreshed

scope, storage scope, lightpen, tablet, etc. Also, the use of coloured graphics facility could greatly improve the quality of ERD/EVD displays and enterprise analysis.

WORKING LANGUAGE OPTIONS

At present, DASS menus, messages/diagnostics, promptings, etc., use English as the working language. It is possible to get around the language limitation as follows:

- o store menus, messages etc., on the DASS control database in different languages
- o ask the user to pre-select the working language at DASS invocation time

ERD VALIDATION

The current implementation of DASS does not validate the ERD components when they are drawn/updated. A valuable enhancement would be an optional ERD validation feature that checks the ERD being drawn against the data in the meta-data database for consistency.

DIRECT GENERATION OF ERD'S

Technically, it is possible to generate ERD's for a given enterprise, directly from the information residing on the meta-data database. It is likely that such

automatically generated ERD's might appear to be aesthetically less pleasing. However, suitable heuristic methods could be developed to get a 'first-cut' ERD for an enterprise, which could then be re-cycled by the user on a need basis.

EVD REFINEMENT

DASS provides three levels of abstraction for enterprise viewing: global, contextual and local. At the lowest (local) level, one part of the view consists of an ERD. A valuable extension is a further step of view refinement where a graphical cum textual description of entities/relationships could be provided. The view refinement could also be taken one step ahead, to the atomic level, where graphical cum textual descriptions of the attributes could be provided as well.

ENTERPRISE VIEW ENHANCEMENT

During enterprise viewing, the EVDs at each conceptual level should provide a pointer to the associated input/output stimuli data. This can facilitate checking of the input-process-output validity/stability at each enterprise/sub-enterprise level.

ERD/EVD CHART GENERATION INTERFACE

The enterprise viewer facility of DASS offers the capability for defining ERD/EVD charts (recall that an ERD/EVD chart is a matrix of ERD/EVD segments). The plotting of these ERD/EVD charts is a valuable extension to DASS. The graphics plotter is part of the current hardware configuration. What is required in future is an interface between the graphics terminal and plotter for generating the plotter output.

USER SESSION MONITORING/LOGGING

There is a need for monitoring/logging individual session activities on a log database. The purpose of capturing such historical data is twofold: performance analysis and process control. The accounting information provided to the user upon session termination should be enhanced to include finer details - the objective here being to make the user aware of the use/abuse of system resources.

SUBSETTING OF DASS

The data administration functions in most enterprises are largely non-centralized. Further, personal computers are becoming more and more popular among end-users. It is therefore advantageous to subset DASS, so that only a desired portion of it is enabled to operate on a

micro-computer.

MAN-MACHINE COMMUNICATION

There is a need for treating human engineering issues seriouly and with deliberation in the design of state of the art software products [Ehrich,1982]. Though, DASS has been designed with due regard to human engineering aspects, it is worth reviewing the entire area of system-user interface. The review should address such issues as: terminology, language interface, menu display and selection, error recovery, etc. It is an excellent idea to involve end-user representatives in these reviews - such participation could make software building process more responsive to end-user needs. There is more than a subtle difference between the design philosophies of a DASS-like system and its counterparts in other disciplines. A data administration system design has many implications on the human-engineering side. This is because a data administration tool has to be sound for enterprise-wide usage and it should be sensitive enough to cater to the needs of a diverse user community (from the president down to the end-user). Such requirements are almost a non-issue for existing tools in other disciplines.

6.1.2 DASS CONCLUDING REMARKS

The foregoing list of potential enhancements highlight

the future directions of DASS development. Develpment of every software tool is influenced by the state-of-the-art technology and adapted methodology, that is current at the time of its design. Software tools, therefore, have to go through an initial period of continual change. DASS, with its overall structure already in place, is well geared to this evolutionay transition.

REFERENCES

The following abbreviations are used:

AFIPS - American Federation for Information processing
ACM - Association for Computing Machinery
CACM - Communications of the ACM
DB - Data Base (Database)
ERA - Entity -Relationship Approach (Conference)
IEEE - Institute of Electrical and Electronic Engineers
IFIP - Internat'l Federation for Information Processing
ISA - Infotech State-of-the-Art (Reports)
NBS - National Bureau of Standards - U.S. Dept. of Commerce
SIGMOD - Special Interest Group on Management of Data
TODS - Transactions on Database Systems
VLDB - Very Large Data Bases (Conference)

Adiba, M., Delobel, C., Leonard, M. [1976]
'A Unified Approach to Modelling Data in Logical Data Base Design' - Modelling in DB management, North-Holland, 1976, pp.311-338.

Aho, A.V. [1979]
'Principles of Compiler Design' - Addison-Wessley, 1979.

ANSI, ANSI/SPARC [1975]
'Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems' - SIGMOD Bulletin, Vol.7-2, 1976.

Anthony, R.N. [1965]
'Planning & Control Systems: A Framework for Analysis' - Harvard University, 1965.

Appleton, D.S. [1977]
'What DB is not' - Datamation, January 1977, pp.85-92.

Atre, S. [1980]
'Data Base: Structured Techniques for Design, Performance, and Management' - John Wiley, 1980.

Bachman, C.W. [1969]
'Data Structure Diagrams' - Database, vol.2, 1969, pp.4-10.

Bachman, C.W. & Daya, M. [1977]
'The Role Concept in Data Models' - VLDB Conf. Proc., 1977, pp.464-476.

Batini, C. & Santucci, G. [1979]
'Top-down Design in the Entity-Relationship Model' - ERA Conf. Proc., 1979, pp.323-338.

Bech, L.L. [1976]
'An Approach to the Creation of Structured Data processing Systems' - ACM SIGMOD Conf. Proc., 1976. pp.179-188

Benci, E., Bodart, F., Bogaert, H. & Cabanes, A. [1976]
'Concepts for Design of a Conceptual Schema' - Modelling in DB Management, North-Holland, 1976, pp.181-200.

Berild, S. & Nachmens, S. [1977]
'Some Practical Applications of CS4 - DBMS for Associative Databases' - Architecture & models in Database Management Systems, North-Holland, 1977, pp.213-235.

Bertrand, C.P. & Daudenarde, J.J. [1980]
'USAGE: Generating Interactive Application Programs from Grammatical Descriptions' - Database, Vol.11-#3, 1980, pp.76-83.

BICS [1982]
'Business Systems Planning & Business Information Control Study: A Comparison' - IBM Systems Journal, Vol.21-#1, 1982, pp.31-53.

Biller, H. & Neuhold, E.J. [1977]
'Concepts for the Conceptual Schema' - Architecture and Models in Database Systems, North-Holland, 1977, pp.1-30.

Birkhoff, G. & Bartee, T.C. [1970]
'Modern Applied Algebra' - McGraw-Hill, 1970.

Bracci, H.G. & Pelagatti, G. [1976]
'Binary Logical Associations in Data Modelling' - Modelling in Database Management Systems, North-Holland, 1976, pp.125-148.

Brown, A.P. [1978]
'The Next Five Years of DB - a Vendor's viewpoint' - ISA Report on DB Technology, Vol.II, 1978, pp.39-51.

BSP [1981]
'Business System Planning [BSP]' - IBM Corporation, GE20-0527-2, July 1981:

Bubenko, J.A. [1977a]
'An Inferential Abstract Modelling Approach [IAM] to Design of Conceptual Schema' - ACM SIGMOD Conf. Proc., 1977, pp.62-74.

Bubenko, J.A. [1977b]
'Validity & Verification Aspects' of Information Modelling' - VLDB Conf. Proc., 1977, pp.556-565.

Bubenko, J.R. & Yao, B. [1978]
'Database Design Tools' - VLDB Conf. Proc., 1978, pp.2-2.

Bubenko, J.A. [1980]
'Information Modelling in the Context of System Development' - Information Processing 80, North-Holland, 1980, pp.395-411.

Buchmann, A.P. & Dale, A.G. [1979]
'Evaluation Criteria for Logical DB Design Methodologies' - Computer Aided Design, may 1979, pp.121-126.

Cardenas, A.F. & Seeley, R.W. [1975]
'A Simple Data Structure for Internal Graphic Design/Drafting' - The Computer Journal, February 1975, pp.30-32.

Cardenas, A.F. [1979]
'Database Management Systems' - Allyn & Bacon, 1979.

Chamberlin, D.D. [1976a]
'Relational DB Management Systems' - ACM Computing Survey, March 1976, pp.43-66.

Chamberlin et al. [1976b]
'SEQUEL2 - a Unified Approach to Data Definition, Manipulation and Control' - IBM Jornal of Research and Development, November, 1976, pp.560-575.

Chand, D.R. & Surya, B.Y. [1980]
'Logical Construction of Software' - CACM, October 1980, pp.546-555.

Chan, E.P. & Lochovsky, F.H. [1979]
'A Graphical DB Design Aid using the Entity-Relationship Model' - ERA Conf. Proc., 1979, pp.295-310.

Chen, P.P. [1976]
'Entity-relationship Model - Toward a Unified View of Data' - TODS, Vol.1, #1, March 1976, pp.9-36.

Chen, P.P. [1977a]
'The Entity-Relationship Model - a basis for Enterprise View of Data' - AFIPS Conf. Proc., 1977, pp.158-165.

Chen, P.P. [1977b]
'The Entity-Relationship Approach to Logical Database Design' - QED Monograph, 1977.

Chen, P.P. [1979a]
'Recent literature of the E-R Approach' - ERA Conf. Proc., 1979, pp.3-12.

Chen, P.P. & Yao, B.S. [1979b]
'Design & Performance Tools for Database Systems' - VLDB Conf. Proc., 1979, pp.3-15.

Chiang, T.C. & Bergeron, R.F. [1979]
'A Database Management System with an E-R Conceptual Model' - ERA Conf. Proc., 1979, pp.467-476.

Clemons, E.K. [1979]
'Design of a Prototype ANSI/SPARC Database' - AFIPS Conf. Proc., 1979, pp.689-696.

Clemons, E.K. [1981]
'Design of an External Schema Facility to Define & Process Recursive Structures' - TODS, Vol.6-#2, June 1981, pp.295-311.

CODASYL [1978]
'CODASYL Data Description Language Journal of Development' - Material Data Management, Dept. of Supply & Services, Ottawa, 1978.

Codd, E.F. [1982a]
'Relational Database: a Practical Foundatation to Productivity' - CACM, Vol.25-#12, February 1982, pp.109-117.

Codd, E.F. [1982b]
'SQL/DS - A Database Newsletter Interview with E. F. Codd' - Database Newsletter, Vol.10-#3, May 1982, pp.3-6.

Control Data Corporation [1980]
'FORTRAN Version.5 Reference Manual' - Control Data, #60481300, 1980.

Date, C.J. [1977]
'An Introduction to Data Base Systems' - Addison-Wesley, 1977.

Davenport, R.A. [1978]
'Data Analysis for Database Design' - The Australian Journal, Vol.10-#4, November 1978, pp.122-137.

Davenport, R.A. [1979a]
'Logical Database Design - from Entity Model to DBMS Structure' - Australian Computer Journal, Vol.11-#3, August 1979, pp.82-97.

Davenport, R.A. [1979b]
'The Application of Data Analysis - Experience with Entity-Relationship Approach' - ERA Conf. Proc., 1979, pp.603-621.

Davenport, R.A. [1980]
'Data Administration - the need for a new function' - Information Processing 80, North-Holland, 1980, pp.505-510.

Davies, G.B. [1982]
'Strategies for Information Requirement Determination' - IBM Systems Journal, Vol.21-#1, 1982, pp.4-30.

DBDA [1975]
'Data Base Design Aid [DBDA]' - IBM Corporation, GH20-1626-0, 1975.

De Blasis, J-P & Johnson, T.H. [1977]
'Database Administration - Classical patterns, some experiences and trends' - AFIPS Conf. Proc., 1977.

De Marco, T. [1978]
'Structured Analysis & Systems Specification' - Yourdon Press, 1978.

Demers, R.A. [1981]
'System Design for Usability' - CACM, August 1981, pp.494-501.

Eastman, C. Lividini, J. & Stoker, D. [1975]
'A Database for Designing Large Physical Systems' - AFIPS Conf. Proc., 1975, pp.603-611.

Ehrensberger, M. [1976]
'Data Dictionary - A more on the Impossible Dream' - AFIPS Conf. Proc., 1976, pp.9-11.

Ehrich, R.W. & Hartson, H.R. [1982]
'On Effective Software Development Methodology' - ACM Forum, CACM, May 1982, pp.350-350.

Falkenberg, E. [1976]
'Concepts for Modelling Information' - Modelling in DB management Systems, North-Holland, 1976, pp.95-109.

Falkenberg, E.D. [1978]
'Data Models: The Next Five Years' - ISA Report on DB Technology, Vol.II, 1978, pp.55-67.

Finkelston, J. & Toms, F. [1980]
'Shared Data Structures' - GUIDE-51 Conf. Proc., 1980, pp.1851-1860.

Finkelston, J. & Wertz, C. [1980]
'Data Administration Concepts' - GUIDE-51 Conf. Proc., 1980, pp.1841-1850.

Finkelstein, C. [1981]
'The Evolution of Information Engineering Methodologies' - Computerworld, May 1981.

Finneran, T.R. & Henry, J.S. [1977]
'Structured Analysis for Database Design' - Datamation, November 1977, pp.102-113.

Flavin, M. [1981]
'Fundamental Concepts of Information Modelling' - Yourdon Press, New York, 1981.

Florentin, J.J. [1976]
'Database Representation of Application Models' - Computer Journal, February 1976, pp.13-16.

Flynn, R.L. [1974]
'A Brief History of DB Management' - Datamation, August 1974, pp.71-77.

Fry, P.J. & Sibley, H.E. [1976]
'Evaluation of DB Management Systems' - ACM Computer Surveys, March 1976, pp.7-42.

Gambino, T.J. & Gerritsen, R. [1977]
'A DBatabase Design Decision Support System' - VLDB Conf. Proc., 1977, pp.534-544.

Gane, C. P. & Sarson, T. [1977]
'Structured System Analysis: Tools & Techniques' - Prentice-Hall, 1979.

Gillenson, M.L. [1982]
'The State of Practice of Data Administration - 1981' - CACM, October 1982, pp.699-706.

Grerritsen, R. [1975]
'A Preliminary System for the Design of DBTG Data Models' - CACM, October 1975, pp.551-557.

Guttag, J. [1977]
'Abstract Data Types & Development of Data Structures' - CACM, June 1977, pp.396-404.

Haseman, W. & Whinston, A.B. [1976]
'Introduction to Data Management' - Richard-Irwin, 1976.

Haseman, W. [1977]
'GPLAN: An Operational DSS' - Database, Vol.8-#3, 1977, pp.73-78.

Hawryszkiewycz, I.T. [1980]
'Data Analysis - What are the Necessary Concepts' - The Australian Computer Journal, February 1980, pp.2-14.

Hawryszkiewycz, I.T. [1981]
'Some Trends in System Design Methodologies' - The Australian Computer Journal, February 1981, pp.13-23.

Head, R.V. [1971]
'Automated System Analysis' - Datamation, August 1971, pp.22-24.

Holland, R.H. [1980]
'DBMS: Developing User Views' - Datamation, February 1980, pp.141-144.

Holland, R.H. [1982]
'Data Base Stability' - Holland Systems Corporation, Ann Arbor, 1982

Housel, B.C., Waddle, V. & Yao, S.b. [1979]
'The Functional Dependency Model for Logical Database Design' - VLDB Conf. Proc., 1979, pp.194-208.

Howden, W.E. [1982]
'Contemporary Software Development Environments' - CACM, May 1982, pp.318-329.

Hsu, J. & Rousopoulos, R. [1979]
'Database Conceptual Modelling' - ERA Conf. Proc., 1979, pp.259-275.

Hubbard, G.U. & Raver, N. [1975]
'Automatic Logical File Design' - VLDB Conf. Proc., 1975, pp.227-253.

Hubbard, G.U. [1979]
'Computer Assisted Logical Database Design' - Computer Aided Design, May 1979, pp.169-179.

Hubbard, G.U. [1981]
'Computer Assisted Database Design' - Van Nostrand Reinhold, 1981.

IBM Corporation [1975]
'Data Base Design Aid [DBDA]' - IBM Corporation, GH20-1626-0, 1975.

IBM Corporation [1981]
'Business System Planning' - IBM Corporation, GE20-0527-2, July 1981.

Iossiphidis, J. [1979]
'A Translator to Convert the DDL of ERM to DDL of System 2000' - ERA Conf. Proc., 1979, pp.477-503.

Irani, K.B., Purkayastha, S. & Teory, T.J. [1979]
'A Designer for DBMS Processable Logical DB Structures' - VLDB Conf. Proc., 1979, pp.219-231.

Jackson, M.A. [1975]
'Principles of Program Design' - Academic Press, 1975.

Kebel, K.N. & Morling, C.R. [1980]
'Interactive Program Generator for IMS Applications' - Database, Vol.11-#3, 1980, pp.35-39.

Kent, W. [1976]
'New Criteria for Conceptual Model' - Large Database Systems, North-Holland, 1976, pp.1-12.

Kent, W. [1977]
'Entities & Relationships in Information' - Architecture & Models in DB Management Systems, North-holland, 1977, pp.67-91.

Kent, W. [1978]
'Data and Reality' - North-Holland, 1978.

Kerschberg, L., Klug, A. & Tsichritzis, D. [1976]
'Ataxonomy of Data Models' - Systems for Large Databases, north-holland, 1976, pp.43-64.

Khan, B.K. [1976]
'A Method for Describing Information Required by the DB Design Process' - ACM SISIGMOD Conf. Proc., 1976, pp.53-64.

Konsynski, B.R. & Mannino, M. [1979]
'Information Resource Specification & Design Language' - ERA Conf. Proc., 1979, pp.339-351.

Langefors, B. [1980]
'Infological Models and Information User Views' - Info Systems, Vol.5, 1980, pp.17-32.

Lawrence, M.J. [1979]
'The Computer Database Decision' - The Australian Computer Journal, February 1979, pp.13-20.

Ledgard, H.F. & Taylor, R.W. [1977]
'Selected Papers from Conference on Data: Abstraction, Definition & Structure' - CACM, June 1977, pp.382-384.

Lefkovits, H.C. [1977]
'Data Dictionary Systems' - QED Information Sciences, 1977.

Leong-Hong, B. & Marron, B. [1978a]
'Database Administration:Concepts, Tools, Experiences, and Problems' - U.S. Dept. of Commerce Publications, 1978.

Leong-Hong, B. & Marron, B. [1978b]
'Technical Profiles of seven Data Element Dictionary&/Directory Systems' - U.S. Dept. of Commerce Publications, 1978.

Liberman, A.Z. [1979]
'Mechanized Analysis of Enttity-relationship design Databases' - ERA Conf. Proc., 1979, pp.641-663.

Ling, R.F. [1980]
'General Considerations on the Design of an Interactive System for Data Analysis' - CACM, March 1980, pp.147-154.

Lusk, E.L. & Overbeek, R.A. [1979]
'A Data Model for Entity-Relationship Models' - ERA Conf. Proc., 1979, pp.445-461.

Machgeels, C. [1976]
'A Procedural Language for Expressing Integrity - constraints in the co-existence Model' - Database management Systems, North-holland, 1976, pp.293-301.

Malkin, J.G. & Anderson, B.F. [1978]
'A Case Management System: Three Views of an Application' - ACM SIGMOD Conf. Proc., 1978, pp.89-100.

Manola, F. [1978]
'An Evaluation of New CODASYL & ANSI/SPARC Database Proposals' - ISA Report on Database Technology, Vol.II, 1978, pp.133-150.

Martin, J. [1975]
'Computer Database Organization' - Prentice-Hall, 1975.

Martin, J. [1980]
'Managing the Database Environment - Vol.I & II' - Savant Institute,1980

Martin, J. [1981]
'Stategic System Planning' - Savant Institute, 1981.

Martin, J. & Finkelstein, C. [1981]
'Information Engineering Vol.I & II' - Savant Institute, 1981.

McCarthy, W.E. [1979]
'Construction & Use of Integrated Accounting Systems with Entity-Relationship Modelling' - ERA Conf. Proc., 1979, pp.625-637.

Meltzer, H.S. [1976]
'Structure & Redundancy in the Conceptual Schema' - Systems for Large Databases, North-holland, 1976, pp.13-25.

Mercier, R.H. [1980]
'An Approach to Information System Control' - GUIDE-50 Conf. Proc., May 1980, pp.1379-1388.

Michaels, A.S., Mittman, B. & Carlson, C.R. [1976]
'A Comparison of Relational & CODASYL Approach to Database Management' - ACM Computer Surveys, March 1976, pp.125-151.

Mito[illegible], M.F. & Irani, K.B. [1975]
'Automatic Database Schema Design & Optimization' - VLDB Conf. Proc., 1975, pp.286-321.

Moulin et al. [1976]
'Conceptual Model as a Database Design Tool' - Modelling in Database Management Systems, North-Holland, 1976, pp.221-238.

Navthe, S.B. & Schkolnick, M. [1978]
'View Representation in Logical DB Design' - ACM SIGMOD Conf. Proc., 1978, pp.144-156.

Navthe, S.B. & Lemke, J. [1979]
'On the Implementation of Conceptual Schema Model within a Three-Level DBMS Architecture' - AFIPS Conf. Proc., Vol.48, 1979, pp.697-708.

NBS [1978]
'Technical Profile of Seven Data Element Dictionary/Directory Systems' - NBS, U.S. Dept. of Commerce, #SP-500-3, 1978.

Newman, P.S. [1982]
'Towards an Integrated Development Environment' - IBM Systems Journal, Vol.21-#1, 1982, pp.81-107.

Newman, W.M. & Sproull, R.F. [1979]
'Principles of Interactive Computer Graphics' - McGraw-Hill Book Co., 1979.

Nijssen, G.M. [1976]
'A Gross Architecture for Next Generation DB Management Systems' - Modelling in DB Management Systems, North-Holland, 1976, pp.1-24.

Nijssen, G.M. [1978]
'Next Five Years in DB Technology' - ISA Report on DB Technology, Vol.II, 1978, pp.215-256.

Nolan, R.L. [1979]
'Managing the Crises in Data Processing' - Havard Business Review, March-April 1979, pp.115-126.

Nunamaker, J.F. & Konsynski, B.R. [1976]
'Computer Aided Analysis & Design of Information Systems' - CACM, December 1976, pp.674-687.

O'Connel, M.L. [1978]
'The 1978 CODASYL DB Specifications' - ISA Report on Database Technology, Vol.II., 1978, pp.259-269.

Oren, O. [1979]
'Statistics for the Usage of a Conceptual Data Model as a basis for Logical DB Design' - VLDB Conf. Proc., 1979, pp.140-145.

Orr, K.T. [1981]
'Structured Requirement Definition' - Ken Orr Associates Inc., 1981.

Orsey, R.R. [1982]
'Business Systems Planning: Management of Information' - Computer Decisions, February 1982, pp.154-158.

Palmer, I.R. [1975a]
'Data Base Systems - a Practical Reference' - CACI, [1975]

Palmer, I.R. [1975b]
'Review of DB Software' - ISA Report on Database Systems, 1975, pp.459-487.

Parker, M.M. [1982]
'Enterprise Information Analysis: Cost Benefit Analysis & the Data-managed System' - IBM Systems Journal, Vol.21-#1, 1982, pp.108-123.

Piney, C. [1977]
'A Generalized Interactive System for Acquisition of structured Data' - Computer Journal, February 1977, pp.37-44.

Quintella, H.M., Gusmao, G.A. & Oliveira, M.R. [1979]
'A Methodologic Guideline for the Installation of Database Development Tools & Data Administration tools: Two Case Studies from Brazil' - VLDB Conf. Proc., 1979, pp.313-318.

Radhakrishnan, T., D'Cunha, A. & Venkatesh, K. [1978]
'A Graphic Editor for Flowchart-like Pictures' - un-published paper, 1978.

Raver, N. & Hubbard, G.U. [1977]
'Automated Logical DB Design: Concepts & Applications' - IBM System Journal, Vol.16-#3, 1977, pp.287-312.

Robinson, K.A. [1975]
'Relational DB Techniques' - ISA Report on Database Systems, 1975, pp.515-523.

Ross, R.G. [1981]
'Data Dictionaries and Data Administration' - AMACOM, New York, 1981.

Roussopoulos, N. [1979]
'Tools for Designing Conceptual Schemata of Databases' - Computer Aided Design, May 1979, pp.119-120.

Sandberg, G. [1981]
'A primer on Relational DB Concepts' - IBM System Journal, Vol.20-#1, 1981, pp.23-40.

Sakamoto, A. & Ball, F.W. [1982]
'Supporting Business Systems Planning Studies with the DB/DC Data Dictionary' - IBM Systems Journal, Vol.21-#1, 1982, pp.54-80.

Senko, M.E. [1976]
'DIAM as a Detailed Example of the ANSI/SPARC Architecture' - Modelling in DB Management Systems, North-Holland, 1976, pp.73-94.

Senko, M.E. [1977]
'Data Structures, Data Accessing in Database Systems' - Past, Presesent, Future' - IBM System Journal, Vol.16-#3, 1977, pp.208-257.

Senko, M.E. [1978]
'Classification: A Basic Tool for Database & Information Systems' - ISA Report on Database Technology, Vol.II, 1978, pp.303-315.

Shave, M.J. [1981]
'Entities, Functions & Binary Relations: Steps to a Conceptual Schema' - The Computer Journal, Vol.24-#1, 1981, pp.42-47.

Shneiderman, B. [1982]
'Designing Computer Messages' - CACM, September 1982, pp.610-611.

Sibley, E.H. [1976]
'The Development of DB Technology' - ACM Computer Surveys, March 1976, pp.1-5.

Sibley, E.H. [1977]
'The Impact of DB Technology on Business Systems' - IFIP Conf. Proc., 1977, pp.589-596.

Smith, J.M. & Smith, C.P. [1977]
'Data Abstractions: Aggregation' - CACM, June 1977, pp.405-413.

Stamen, J. & Costello, W. [1981]
'Evaluating DB Languages' - Datamation, May 1981, pp.116-122.

Stamper, R. [1973]
'Information in Business & Administrative Sysstems' - John Wiley & Sons, 1973.

Stanfield, S.H. [1980]
'Data as a Resource' - GUIDE-51 Conf. Proc., 1980.

Steel, T.B. [1978]
'The Current ANSI/SPARC Proposals' - ISA Report on DB Technology, Vol.II, 1978, pp.341-355.

Sundgren, B. [1978]
'DB Design in Theory & Practice: Towards an Integrated Methodology' - VLDB Conf. Proc., 1978, pp.3-20.

Synott, W.R. & Grubber, W.H. [1981]
'Information Resource Management' - John-Wiley, 1981.

Tardieu, H., Pascot, D., Nanci, D. & Heckenroth, H. [1979]
'A Method, A Formalism & Tools for Database Design' - ERA Conf. Proc., 1979, pp.353-377.

Taylor, R.W. & Frank, R.L. [1976]
'Codasyl DB Management System' - ACM Computing Surveys, March 1976, pp.67-103.

Teichroew, D. & Sayani, H. [1971]
'Automation of System Building' - Datamation, August 1971, pp.25-30.

Teichroew, D. & Hershey, E.A. [1977]
'PSL/PSA - A Computer Aided Technique for Structured Documentation and analysis of Information Processing Systems' - IEEE Transactions on Software Engineering, January 1977, pp.41-48.

Teory, T.J. & Fry, J.P. [1978]
'Logical DB Design - a Pragmatic Approach' - ISA Report on DB Technology, Vol.II, 1978, pp.358-383.

Teory, T.J. & Fry, J.P. [1980]
'Logical Record Access Approach to DB Design' - ACM Computing Surveys, June 1980, pp.179-211.

Tektronix Inc. [1979]
'Tektronix Plot 10 Terminal Control System - User's Manual' - Tektronix Inc., #062-1474-00, May 1979.

Tozer, E. [1979]
'Database Design Experience' - ERA Conf. Proc., 1979, pp.623-624.

Tsao, J.H. [1979]
'Enterprise Schema - an Approach to IMS Logical DB Design' - ERA Conf. Proc., 1979, pp.585-601.

Tsichritzis, D.C. [1975]
'Features of a Conceptual Schema' - VLDB Conf. Proc., 1975, pp.532-534.

Tsichritzis, D.C. & Lochovsky, F.H. [1976]
'Hierarchical DB Management: A Survey' - ACM Computer Surveys, March 1976, pp.105-123.

Tsichritzis, D.C. & Lochovsky, F.H. [1978]
'Designing the Database' - Datamation, August 1978, pp.147-151.

Tsichritzis, D.C. & Lochovsky, F.H. [1982]
'Data Models' - Prentice-Hall, 1982.

Ullman, J.D. [1980]
'Principles of Database Design' - Computer Science Press, 1980.

UNIVAC [1980a]
'OPTIMA 1100 Project Management System' - Sperry Univac, UA0368, 1980.

UNIVAC [1980b]
'Sperry Univac Series 1100 MAPPER User Reference Guide' - #UP9193, Sperry Univac, 1980.

UNIVAC [1981]
'DMS-1100 System Support Functions' - Sperry Univac, UP7909, 1981.

Vincent, D. [1977]
'The Use of Entity Diagrams in database System Implementation' - ACM 77 Conf. Proc., 1977, pp.62-63.

Warnier, J.D. [1974]
'Logical Construction of Programs' - Van Nostrand Reinhold, 1974.

Weinberg, V. [1978]
'Structured Analysis' - Yourdon, 1978.

Weldon J-L. [1981]
'Data Base Administration', Plenum Press, 1981.

Yeh, R.T., Roussopoulos, n. & Chang, P.Y. [1978]
'DB Design -- An Approach & Some Issues' - ISA Report on DB Technology, Vol.II, 1978, pp.444-448.

Yourdon, E. & Constatine, L.L. [1978]
'Structured Design: Fundamentals of a Discipline of Computer Program System Design' - Yourdon Press, 1978.

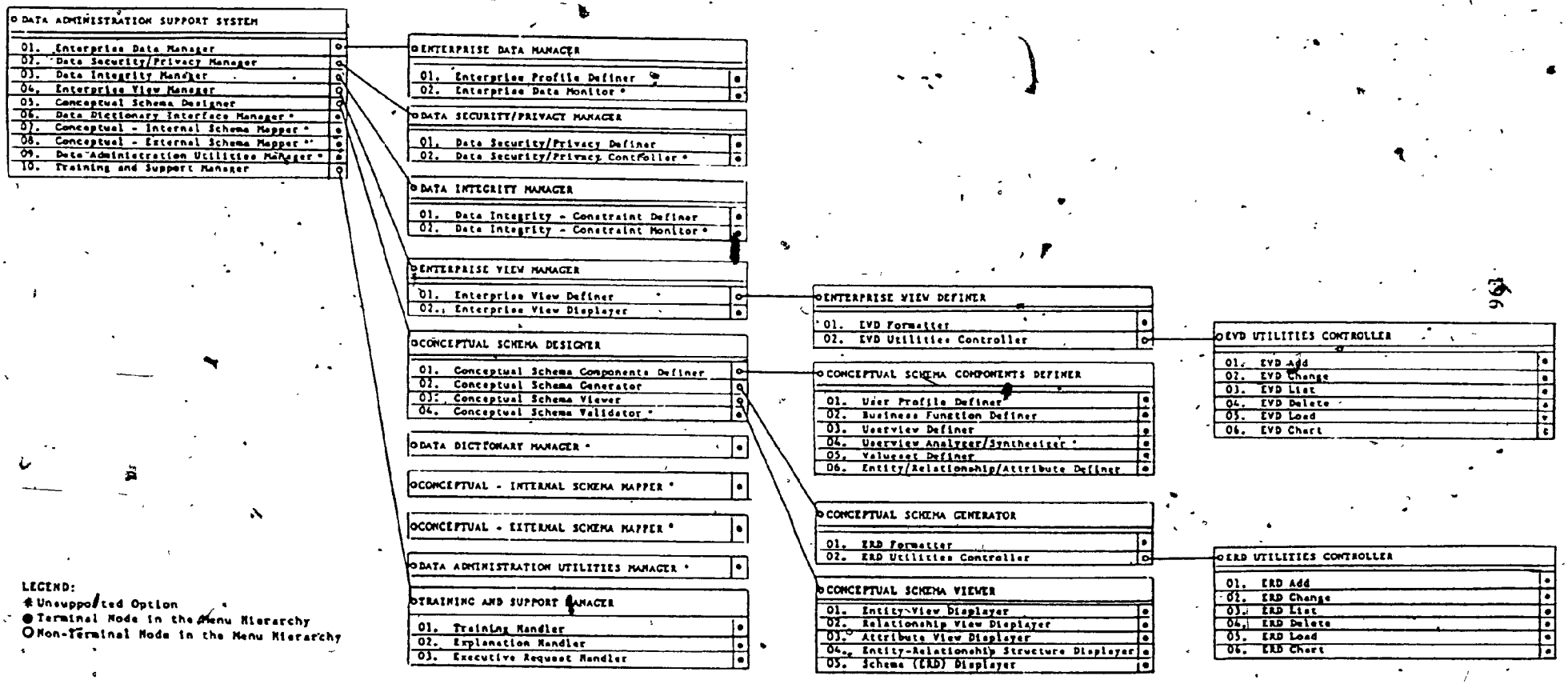
Zachman, J.A. [1982]
'Business Systems Planning & Business Information Control Study: A Comparison' - IBM System Journal, Vol.21-#1, 1982, pp.31-53.

Zimmermann, k. [1975]
'Different Views of a Database: Co-existence between Network model & Relational Model' - VLDB Conf. Proc., 1975, pp.535-537.

Zloof, M.M. [1977]
'Query-By-Example: a Database Language' - IBM Systems Journal, Vol.16-#4, 1977, pp.324-343.

APPENDIX 1

DASS MENU HIERARCHY



Data Administration Support System - Menu Hierarchy

APPENDIX 2

DASS SESSION DEMO

```
call,dass(dbx=abc)

      (. .)
        V
        !
       //:\\
      (   :  )
          :
          :
       // \\
      //   \\  DASS VER.3R1 82/12/30 @ 05.41.27

 DATA ADMIN SUPPORT SYSTEM
 -------------------------
 01. ENTERPRISE DATA MANAGER
 02. DATA SECURITY/PRIVACY MANAGER
 03. DATA INTEGRITY MANAGER
 04. ENTERPRISE VIEW MANAGER
 05. CONCEPTUAL SCHEMA DESIGNER
 06. DATA DICTIONARY INTERFACE MANAGER*
 07. CONCEPTUAL-INTERNAL SCHEMA MAPPER*
 08. CONCEPTUAL-EXTERNAL SCHEMA MAPPER*
 09. DATA ADMINISTRATION UTILITIES MANAGER*
 10. TRAINING AND SUPPORT MANAGER

 <1>  TYPE-IN SELECTION:
? 05

 CONCEPTUAL SCHEMA DESIGNER
 --------------------------
 01. CONCEPTUAL SCHEMA COMPONENT DEFINER
 02. CONCEPTUAL SCHEMA GENERATOR
 03. CONCEPTUAL SCHEMA VIEWER
 04. CONCEPTUAL SCHEMA VALIDATOR*

 <2>  TYPE-IN SELECTION:
? $status

 ECHO=OFF,PLOT=OFF,ACOM=OFF,MDBA=OFF,MGRA=OFF,DBUG=OFF,TEST=OFF
 QUIK=OFF,GROP=0,DBOP=0,ERDF=0


 <2>  TYPE-IN SELECTION:
```

```
? 01
 01

 CONCPTL SCHEMA COMPNT DEFINER
 -----------------------------
 01. USER PROFILE DEFINER
 02. FUNCTION/PROCESS/ACTIVITY DEFINER
 03. USERVIEW DEFINER
 04. USERVIEW ANALYZER/SYNTHESIZER*
 05. VALUESET DEFINER
 06. ENTITY/RELATIONSHIP/ATTRIBUTE DEFINER

 <3>  TYPE-IN SELECTION:
? 06
 06
      ----
 [3]  DASS - ENTITY-RELN-ATTRIB DEFN
      ----


 <3>  TYPE-IN COMMAND:
? list entity(emploe)

 LIST    =ENTD,EMPLOE


 ------
 EMPLOE- LIST
 ------

 - ENTITY NAME :
   EMPLOE
 - ENTITY SYNONYM :
   EMPL
 - SECURITY PASSWORD :
   RXDX
 - ENTITY-ID :
   E1800
 - DESCRIPTION:
   EMPLOYER(EMPLOE)='ABC ENGINEERING INCORPORATED'
   POSTING(EMPLOE)='ABC MARKETING DIVISION'
   STATUS(EMPLOE)=FULL-TIME,PART-TIME
```

```
 - ENTITY-TYPE (DEP/INDEPENDENT) :
   INDEPENDENT
 - ENTITY EXISTENCE PROBABILITY :
   100 %
 - ENTITY SIZE (#CHARS) :
   84
 - ENTITY CARDINALITY :
   450
 - ENTITY EXPECTED GROWTH :
   10 %
 - ENTITY ACCESS FREQUENCY :
   600/DAY
 - ENTITY ACESS VOLUME :
   200/ACCESS
<><><>

COMPLETE..LIST   =ENTD,EMPLOE

<3>  TYPE-IN COMMAND:
? add attr(salary) to entt(emploe)

ADD    =ATTD,SALARY  @ENTD,EMPLOE


------
SALARY- ADD
------

ATTRIBUTE NAME :
SALARY
 SALARY
ATTRIBUTE SYNONYM :
? $set echo=off
$SET ECHO=OFF


SET ECHO=OFF COMPLETE...

ATTRIBUTE SYNONYM :
? $status


ECHO=OFF,PLOT=OFF,ACOM=OFF,MDBA=OFF,MGRA=OFF,DBUG=OFF,TEST=OFF
QUIK=OFF,GROP=0,DBOP=0,ERDF=0
```

```
 ATTRIBUTE SYNONYM :
? empsal
  EMPSAL
 SECURITY PASSWORD :
? rxdx
  RXDX
 ATTRIBUTE-ID :
? a1750
  A1750
 ATTRIBUTE DESCR/DEFN LINE1 :
? status(emploe)=full-time =>
  STATUS(EMPLOE)=FULL-TIME =>
 ATTRIBUTE DESCR/DEFN LINE2 :
? min(salary)=15000
  MIN(SALARY)=15000
 ATTRIBUTE DESCR/DEFN LINE3 :
? max(salary)=100000
  MAX(SALARY)=100000
 OWNER ENITY/RELATIONSHIP NAME :
 EMPLOE
  EMPLOE
 ATTRIBUTE SIZE (#CHARS) :
? 7
  7
 ATTRIBUTE FORMAT/PRECISION :
? 9(4).99
  9(4).99
 ATTRIBUTE VALUESET NAME :
? valsal
  VALSAL
 KEY STATUS (XKE,CKE,PKE..) :
? n/a
  N/A
 DATA SOURCE (INTERNAL,EXTERNAL) :
? external
  EXTERNAL
 INTEGRITY-CONSTR ID. :
? in200
  IN200
 COMPLETE..ADD    =ATTD,SALARY  @ENTD,EMPLOE

 <3>  TYPE-IN COMMAND:
?
  change attr(salary) of entity(emploe)

 CHAN   =ATTD,SALARY  @ENTD,EMPLOE


 ------
 SALARY- CHANGE
 ------
```

```
 ATTRIBUTE NAME :
  SALARY
 *NOTE* - TO CHANGE NAME USE <RENAME> CMND
 ATTRIBUTE SYNONYM :
  EMPSAL
  !
? +05
 ATTRIBUTE DESCR/DEFN LINE3 :
  MAX(SALARY)=100000
  !
? $next
 OWNER ENITY/RELATIONSHIP NAME :
  EMPLOE
  !
? +02
 ATTRIBUTE FORMAT/PRECISION :
  9(4).99
  !
? 9(5).99
 ATTRIBUTE VALUESET NAME :
  VALSAL
  !
? $prev
 ATTRIBUTE FORMAT/PRECISION :
  9(5).99
  !
? $exit
 COMPLETE..CHAN    =ATTD,SALARY  @ENTD,EMPLOE

 <3>  TYPE-IN COMMAND:
? save,t

 SAVE,T - WHAT ?
? $exit

 <3>  TYPE-IN COMMAND:
? $save,t


 SAVE,T    COMPLETE...



 <3>  TYPE-IN COMMAND:
? $print,t 01,03


  1              SALARY
  2              EMPSAL
  3              RXDX
```

```
 <3>  TYPE-IN COMMAND:
?$clear,t


 CLEAR COMPLETE...


 <3>  TYPE-IN COMMAND:
? $print,t 01,03


  1              * *
  2              *
  3              *


 <3>  TYPE-IN COMMAND:
? $restore,t


 RESTORE   COMPLETE...


 <3>  TYPE-IN COMMAND:
? $print,t 01,03


  1              SALARY
  2              EMPSAL
  3              RXDX


 <3>  TYPE-IN COMMAND:
? $clear


 CLEAR COMPLETE...


 <3>  TYPE-IN COMMAND:
? $menu


 [3]  DASS - ENTITY-RELN-ATTRIB DEFN
```

```
 <3>  TYPE-IN COMMAND:
? list attr(salary) of entt(emploe)

LIST    =ATTD,SALARY  @ENTD,EMPLOE


 ------
 SALARY- LIST
 ------

 - ATTRIBUTE NAME :
   SALARY
 - ATTRIBUTE SYNONYM :
   EMPSAL
 - SECURITY PASSWORD :
   RXDX
 - ATTRIBUTE-ID :
   A1750
 - DESCRIPTION:
   STATUS(EMPLOE)=FULL-TIME =>
   MIN(SALARY)=15000
   MAX(SALARY)=100000
 - OWNER ENITY/RELATIONSHIP NAME :
   EMPLOE
 - ATTRIBUTE SIZE (#CHARS) :
   7
 - ATTRIBUTE FORMAT/PRECISION :
   9(5).99
 - ATTRIBUTE VALUESET NAME :
   VALSAL
 - KEY STATUS (XKE,CKE,PKE...) :
   N/A
 - DATA SOURCE (INTERNAL,EXTERNAL) :
   EXTERNAL
 - INTEGRITY-CONSTR ID :
   IN200
<><><>

COMPLETE..LIST .  =ATTD,SALARY  @ENTD,EMPLOE

<3>  TYPE-IN COMMAND:
```

```
? $usage

**************************************************
DASS ACCOUNTING INFORMATION:
---------------------------
START DATE/TIME      ..... : 82/12/30 @ 05.41.27.
CPU USAGE (SECONDS)  ..... : 12
EXIT TYPE            ..... : >>NORMAL
DATABASE CALLS       ..... : 15
GRAPHICS FACILITY CALLS... : 0
USAGE CHARGE         ..... : $ 2.25
FIN DATE/TIME        ..... : 82/12/30 @ 05.53.25.
**************************************************

<3> TYPE-IN COMMAND:
? $exit

CONCPTL SCHEMA COMPNT DEFINER
-----------------------------
01. USER PROFILE DEFINER
02. FUNCTION/PROCESS/ACTIVITY DEFINER
03. USERVIEW DEFINER
04. USERVIEW ANALYZER/SYNTHESIZER*
05. VALUESET DEFINER
06. ENTITY/RELATIONSHIP/ATTRIBUTE DEFINER

<3> TYPE-IN SELECTION:
? $exit

CONCEPTUAL SCHEMA DESIGNER
--------------------------
01. CONCEPTUAL SCHEMA COMPONENT DEFINER
02. CONCEPTUAL SCHEMA GENERATOR
03. CONCEPTUAL SCHEMA VIEWER
04. CONCEPTUAL SCHEMA VALIDATOR*

<2> TYPE-IN SELECTION:
? 03

CONCEPTUAL SCHEMA VIEWER
------------------------
01. ENTITY VIEW DISPLAYER
02. RELATIONSHIP VIEW DISPLAYER
03. ATTRIBUTE VIEW DISPLAYER
04. ENTITY-RELATIONSHIP STRUCTURE DISPLAYER+
```

```
05. SCHEMA (ERD) DISPLAYER

 <3>  TYPE-IN SELECTION:
? 03

      ----
 [3]  DASS - ATTRIBUTE VIEW
      ----


 TYPE-IN ATTRIBUTENAME:
? salary

 ENTITY OR RELATIONSHIP ?
? ent

 ENTITY/RELATIONSHIP NAME:
? emploe

  ..........
  : SALARY :
  ..........

 PROPERTIES:
 ----------
 - ATTRIBUTE NAME :
   SALARY
 - ATTRIBUTE SYNONYM :
   EMPSAL
 - SECURITY PASSWORD :
   RXDX
 - ATTRIBUTE-ID :
   A1750
 - DESCRIPTION:
   STATUS(EMPLOE)=FULL-TIME =>
   MIN(SALARY)=15000
   MAX(SALARY)=100000
 - OWNER ENITY/RELATIONSHIP NAME :
   EMPLOE
 - ATTRIBUTE SIZE (#CHARS) :
   7
 - ATTRIBUTE FORMAT/PRECISION :
   9(5).99
 - ATTRIBUTE VALUESET NAME :
   VALSAL
 - KEY STATUS (XKE,CKE,PKE..) :
   N/A
 - DATA SOURCE (INTERNAL,EXTERNAL) :
   EXTERNAL
 - INTEGRITY-CONSTR ID :
   IN200
```

```
<><><>

TYPE-IN ATTRIBUTENAME:
? $exit


CONCEPTUAL SCHEMA VIEWER
------------------------
01. ENTITY VIEW DISPLAYER
02. RELATIONSHIP VIEW DISPLAYER
03. ATTRIBUTE VIEW DISPLAYER
04. ENTITY-RELATIONSHIP STRUCTURE DISPLAYER+
05. SCHEMA (ERD) DISPLAYER

<3>  TYPE-IN SELECTION:
? $exit

CONCEPTUAL SCHEMA DESIGNER
--------------------------
01. CONCEPTUAL SCHEMA COMPONENT DEFINER
02. CONCEPTUAL SCHEMA GENERATOR
03. CONCEPTUAL SCHEMA VIEWER
04. CONCEPTUAL SCHEMA VALIDATOR*

<2>  TYPE-IN SELECTION:
? $exit

DATA ADMIN SUPPORT SYSTEM
-------------------------
01. ENTERPRISE DATA MANAGER
02. DATA SECURITY/PRIVACY MANAGER
03. DATA INTEGRITY MANAGER
04. ENTERPRISE VIEW MANAGER
05. CONCEPTUAL SCHEMA DESIGNER
06. DATA DICTIONARY INTERFACE MANAGER*
07. CONCEPTUAL-INTERNAL SCHEMA MAPPER*
08. CONCEPTUAL-EXTERNAL SCHEMA MAPPER*
09. DATA ADMINISTRATION UTILITIES MANAGER*
10. TRAINING AND SUPPORT MANAGER

<1>  TYPE-IN SELECTION:
```

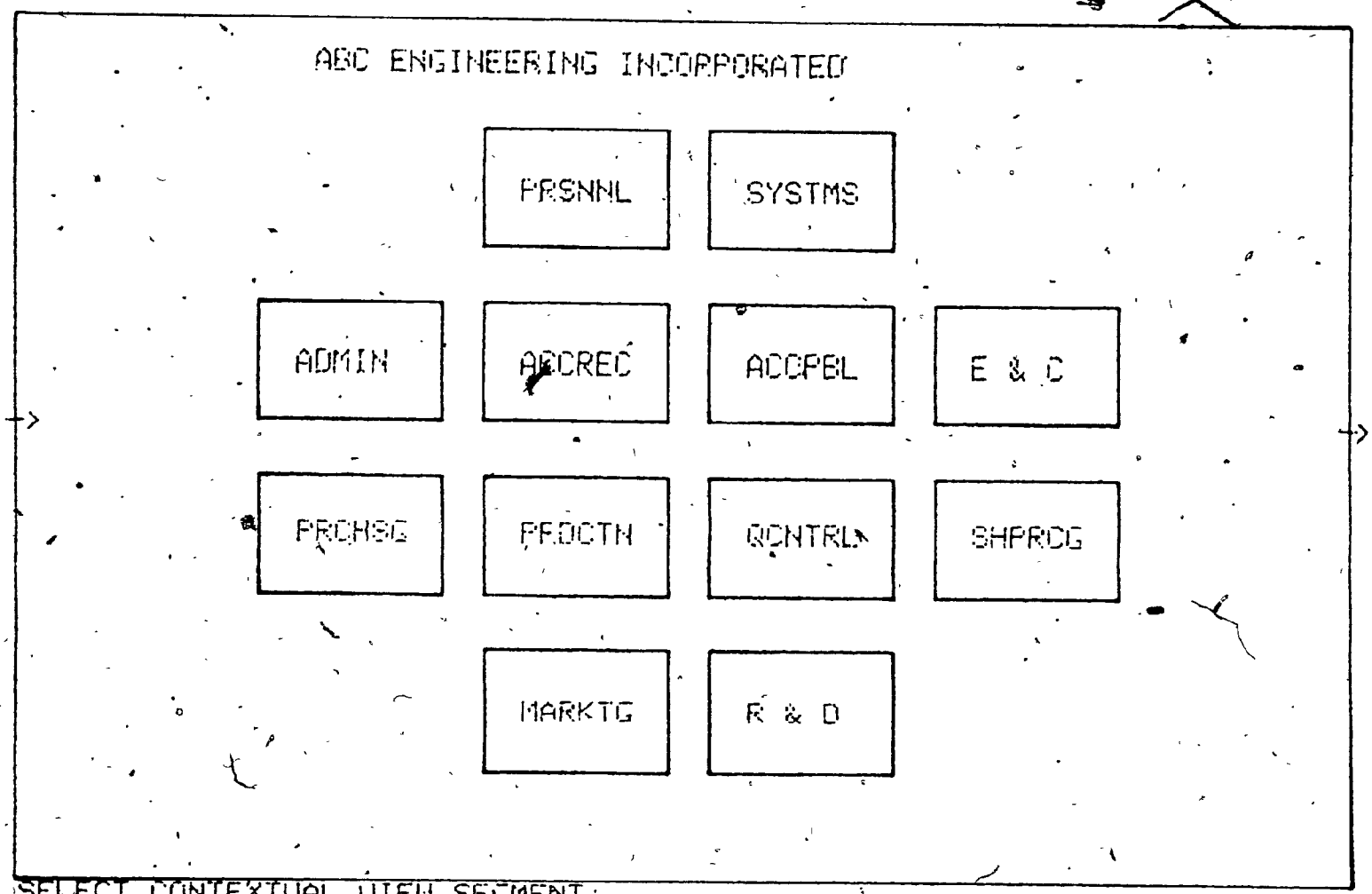
```
? $exit

        (. .)
         V
         !
        //:\\
       (  :  )
          :
          :
          :
        // \\
       //   \\  DASS EXIT ON 82/12/30 @ 06.00.32
```

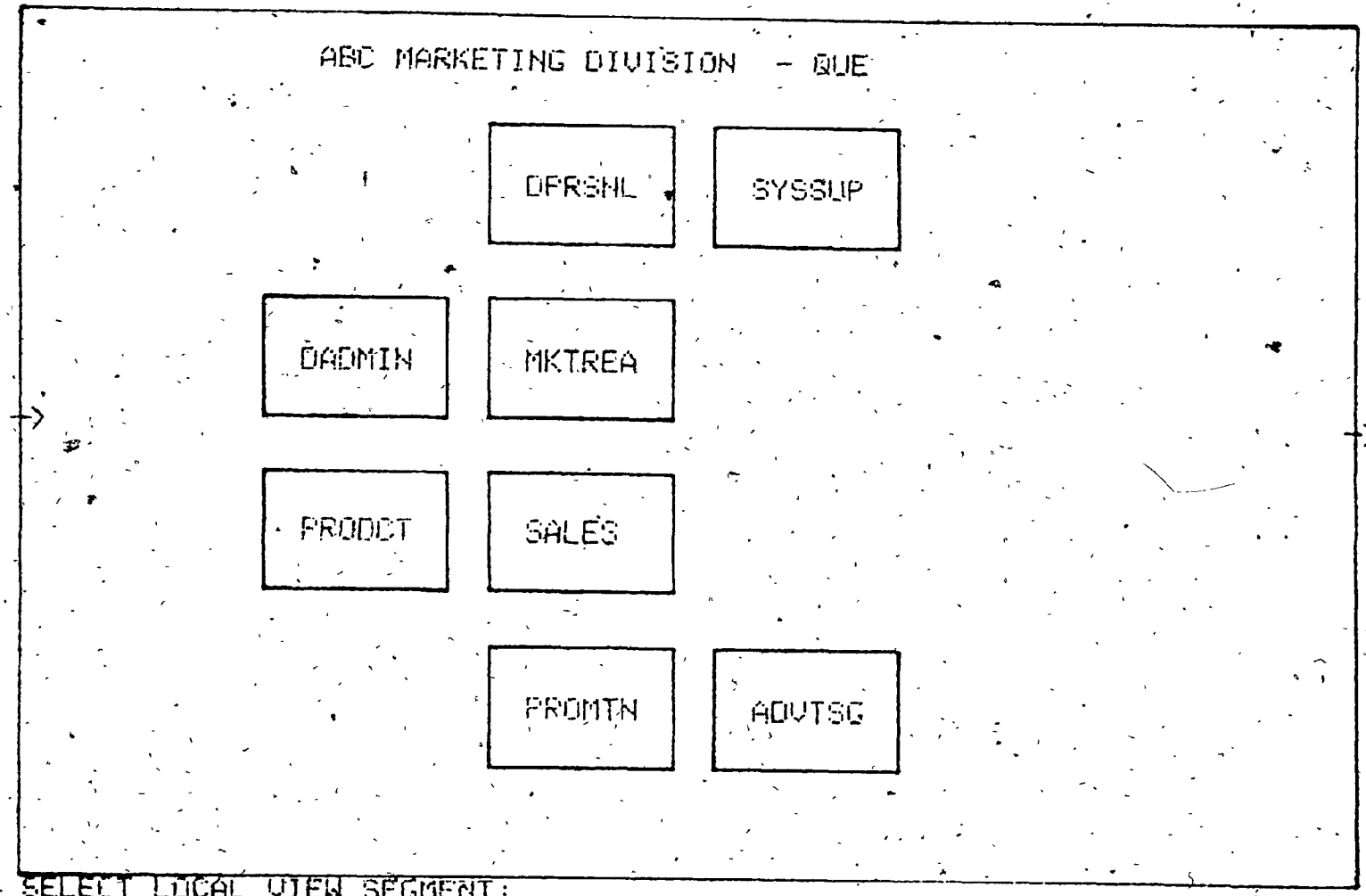
```
**************************************************
DASS ACCOUNTING INFORMATION:
---------------------------
START DATE/TIME      ..... : 82/12/30 @ 05.41.27.
CPU USAGE (SECONDS)  ..... : 12
EXIT TYPE            ..... : >>NORMAL
DATABASE CALLS       ..... : 16
GRAPHICS FACILITY CALLS... : 0
USAGE CHARGE         ..... : $ 2.40
FIN DATE/TIME        ..... : 82/12/30 @ 06.00.32.
**************************************************
```

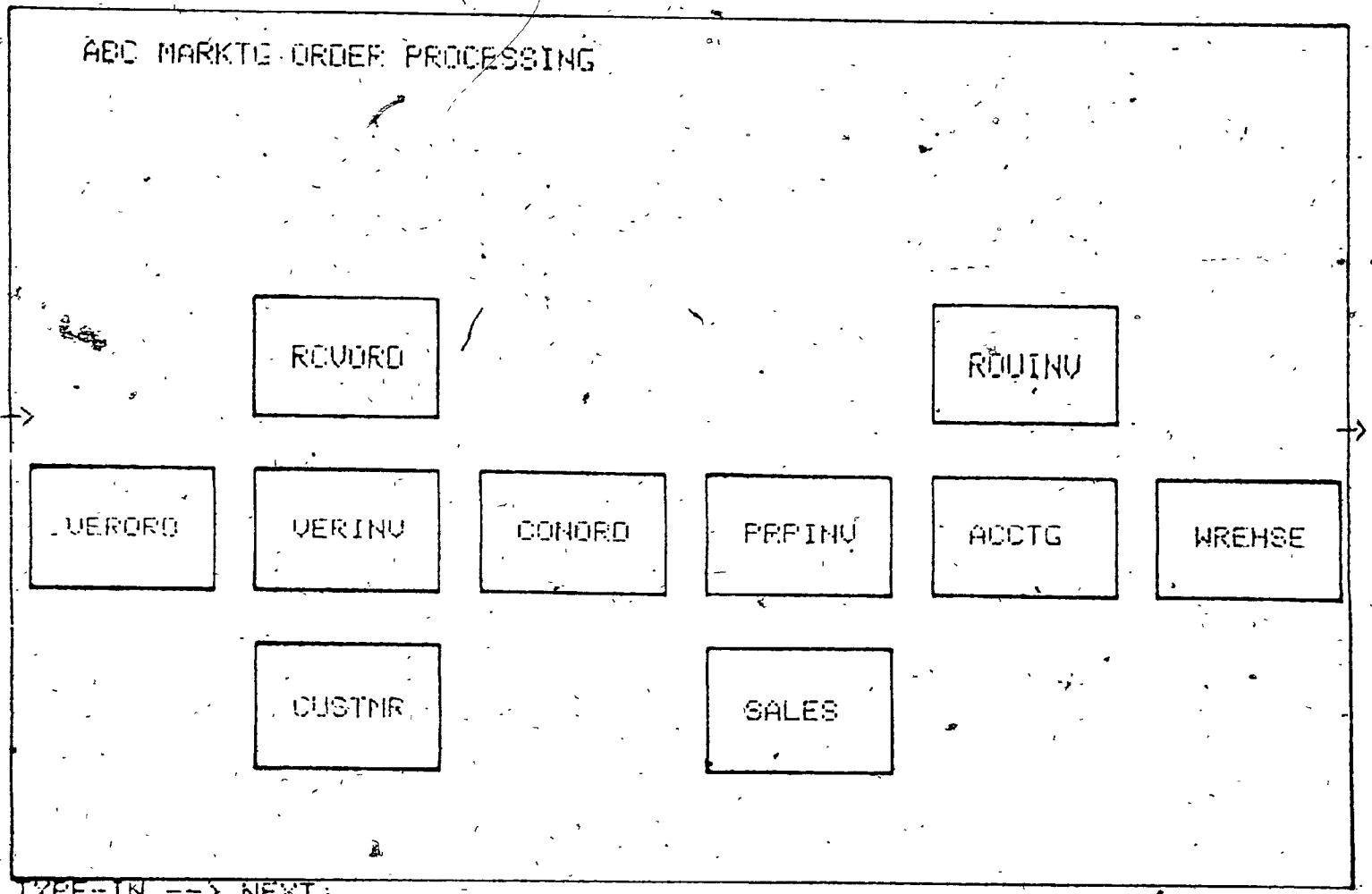
```
*/bye
```

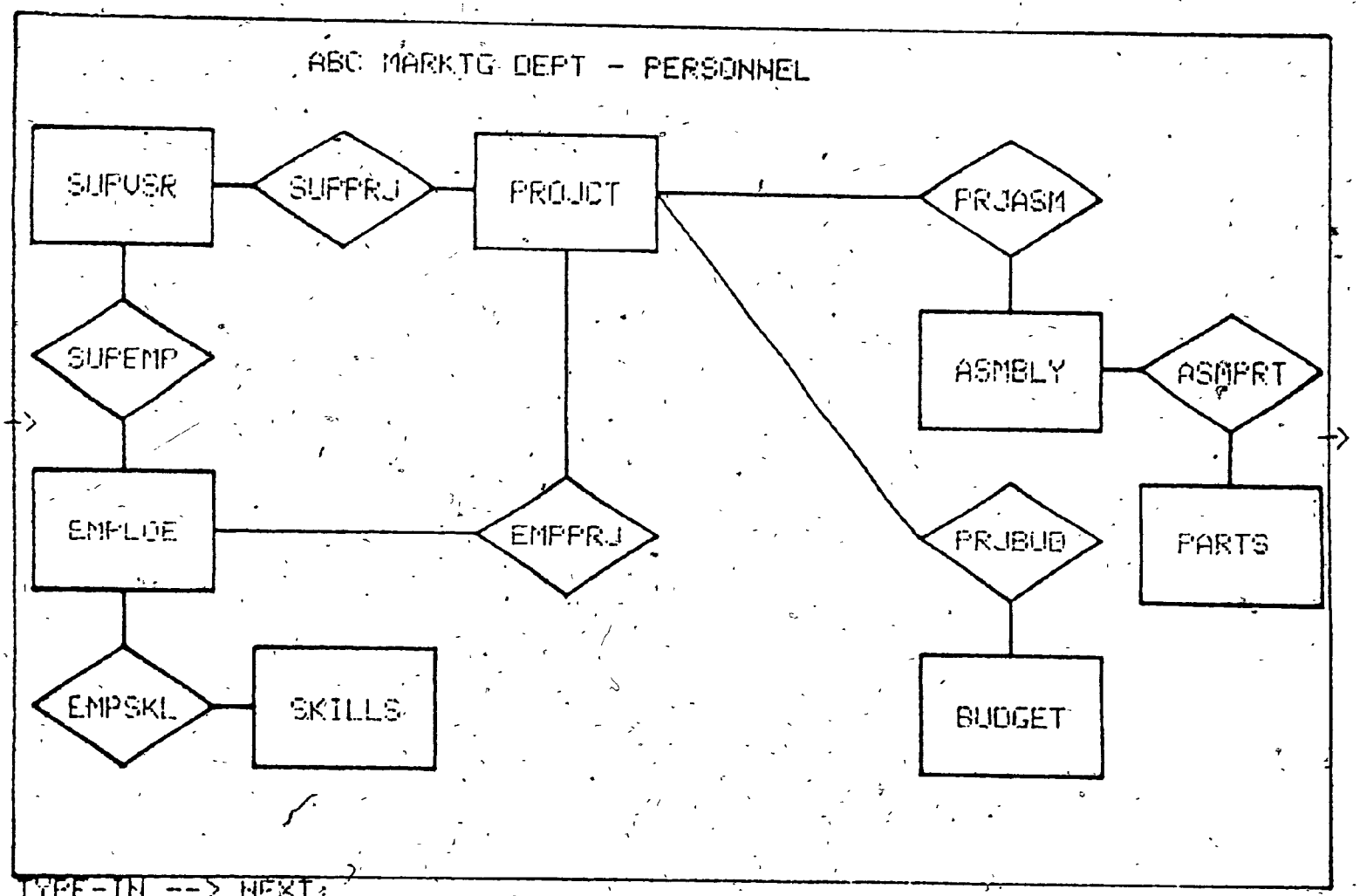
APPENDIX 3

ENTERPRISE VIEWING - STEPWISE VIEW REFINEMENT


ABC ENGINEERING INCORPORATED
PRSNNL
SYSTMS
ADMIN
ACCREC
ACCPBL
E & C
PRCHSG
PRDCTN
QCNTRL
SHPRDG
MARKTG
R & D
SELECT CONTEXTUAL VIEW SEGMENT:


ABC MARKETING DIVISION - QUE
DPRSNL
SYSSUP
DADMIN
MKTREA
PRODCT
SALES
PROMTN
ADVTSG
SELECT LOCAL VIEW SEGMENT:


ABC MARKTG ORDER PROCESSING
RCVORD
RQUINV
VERORD
VERINV
CONORD
PRPINV
ACCTG
WREHSE
CUSTMR
SALES
TYPE-IN --> NEXT:


ABC MARKTG DEPT - PERSONNEL
SUPUSR
SUPPRJ
PROJCT
PRJASM
SUPEMP
ASMBLY
ASMPRT
EMPLOE
EMPPRJ
PRJBUD
PARTS
EMPSKL
SKILLS
BUDGET
TYPE-IN --> NEXT:

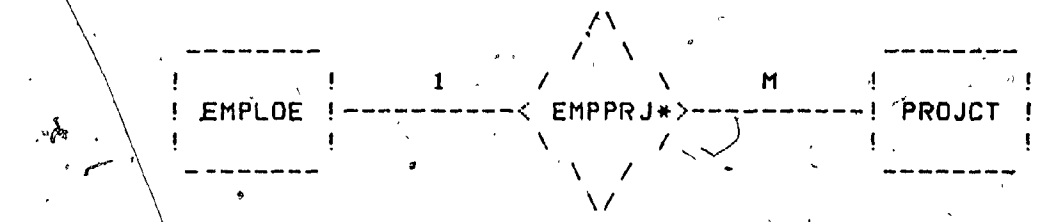
APPENDIX 4

CONCEPTUAL SCHEMA VIEWS

```
      ----
[3]   DASS - ENTITY VIEW
      ----


 TYPE-IN ENTITYNAME:
? emploe

   --------
  !        !
  ! EMPLOE !
  !        !
   --------

 PROPERTIES:
 ----------
 - ENTITY NAME :
   EMPLOE
 - ENTITY SYNONYM :
   EMPL
 - SECURITY PASSWORD :
   RXDX
 - ENTITY-ID :
   E1800
 - DESCRIPTION:
   EMPLOYER(EMPLOE)='ABC ENGINEERING INCORPORATED'
   POSTING(EMPLOE)='ABC MARKETING DIVISION'
   STATUS(EMPLOE)=FULL-TIME,PART-TIME
 - ENTITY-TYPE (DEP/INDEPENDENT) :
   INDEPENDENT
 - ENTITY EXISTENCE PROBABILITY :
   100 %
 - ENTITY SIZE (#CHARS) :
   84
 - ENTITY CARDINALITY :
   450
 - ENTITY EXPECTED GROWTH :
   10 %
 - ENTITY ACCESS FREQUENCY :
   600/DAY
 - ENTITY ACESS VOLUME :
   200/ACCESS
<><><>

 TYPE-IN ENTITYNAME:
? $exit
```

[3] DASS - RELATIONSHIP VIEW

TYPE-IN RELATIONSHIP NAME:
? empprj



PROPERTIES:
- RELATIONSHIP NAME :
  EMPPRJ
- RELATIONSHIP SYNONYM :
  EMPR
- SECURITY PASSWORD :
  EXDX
- RELATION-ID :
  R1200
- DESCRIPTION:
  MEMBERS(EMPPRJ) = EMPLOE,PROJCT
  STATUS(EMPLOE) = FULL-TIME
  STATUS(PROJCT) = DEVELOPMENT
- RELATION-TYPE (1-M,M-M, ...) :
  1-M
- RELATIONSHIP EXISTENCE PROBABILITY :
  75 %
- RELATIONSHIP SIZE (BYTES) :
  90
- RELATIONSHIP CARDINALITY :
  6000
- RELATIONSHIP EXPECTED GROWTH :
  10 %
- RELATIONSHIP ACCESS FREQUENCY :
  600/DAY
- RELATIONSHIP ACESS VOLUME :
  5/ACCESS
- RELATIONSHIP MEMBER ENTITIES :
  EMPLOE,PROJCT
- RELATIONSHIP AFFINITY (AUTOMATIC,CONDITIONAL) :
  CONDITIONAL
<><><>

TYPE-IN RELATIONSHIP NAME:
? $exit

[3] DASS - ATTRIBUTE VIEW

TYPE-IN ATTRIBUTENAME:
? salary

ENTITY OR RELATIONSHIP ?
? ent

ENTITY/RELATIONSHIP NAME:
? emploe

```
  ..........
  : SALARY :
  ..........
```

PROPERTIES:

- ATTRIBUTE NAME :
  SALARY
- ATTRIBUTE SYNONYM :
  EMPSAL
- SECURITY PASSWORD :
  RXDX
- ATTRIBUTE-ID :
  A1750
- DESCRIPTION:
  STATUS(EMPLOE)=FULL-TIME =>
  MIN(SALARY)=15000
  MAX(SALARY)=100000
- OWNER ENITY/RELATIONSHIP NAME :
  EMPLOE
- ATTRIBUTE SIZE (#CHARS) :
  7
- ATTRIBUTE FORMAT/PRECISION :
  9(5).99
- ATTRIBUTE VALUESET NAME :
  VALSAL
- KEY STATUS (XKE,CKE,PKE..) :
  N/A
- DATA SOURCE (INTERNAL,EXTERNAL) :
  EXTERNAL
- INTEGRITY-CONSTR ID :
  IN200

<><><>

TYPE-IN ATTRIBUTENAME:
? $exit

```
[3]  DASS - ENTITY-RELATIONSHIP VIEW

TYPE-IN ENTITY OR RELATIONSHIP NAME:

? empprj
 ENTITY OR RELATIONSHIP ?

? rel

 RELATIONSHIP: EMPPRJ(EMPLOE,PROJCT)

 -------------------------
 !@EMPLNO!.NAME  !.SALARY!
 -------------------------
     !
     !
     !
     !
     V
     V
 ------------------------------------------------
 !@PROJCD!.PNAME !.PSTDAT!.PENDAT!.PPRTY !.PACCT !
 ------------------------------------------------

 LEGEND: @=XKEY,+=CKEY,*=PKEY,.=NKEY
 <><><>
 TYPE-IN ENTITY OR RELATIONSHIP NAME:

? $exit
```

APPENDIX 5

DASS HELP FACILITY

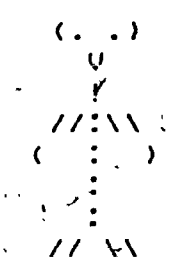
```
$help

TRAINING AND SUPPORT MANAGER
----------------------------
01. TRAINING
02. EXPLANATION
03. EXECUTIVE REQUEST

<0>  TYPE-IN <<HELP>> SELECTION:
? 02


TYPE-IN COMMANDNAME OR ERRNUM:
? usage
  $USAGE                    : $USAGE

* FOR ADDITIONAL INFO SEE <DASS> USERGUIDE...


TYPE-IN COMMANDNAME OR ERRNUM:
? promote
  PRO <WHAT> <WHERE> <HOW>: PRO ATTR(X) OF ENTT(A) AS ENTT(B)

* FOR ADDITIONAL INFO SEE <DASS> USERGUIDE...


TYPE-IN COMMANDNAME OR ERRNUM:
? in1090
  IN1090 - INVALID COMMAND INPUT.....
         - RE-TYPE COMMAND CORRECTLY
* FOR ADDITIONAL INFO SEE <DASS> USERGUIDE...


TYPE-IN COMMANDNAME OR ERRNUM:
? db8010
  DB8010 - DATABASE ACCESS ERROR.........
         - CHECK/INTERPRET RETURN CODE
* FOR ADDITIONAL INFO SEE <DASS> USERGUIDE...


TYPE-IN COMMANDNAME OR ERRNUM:
? $exit
TRAINING AND SUPPORT MANAGER
----------------------------
01. TRAINING
02. EXPLANATION
03. EXECUTIVE REQUEST
```

```
 <0>  TYPE-IN <<HELP>> SELECTION:
? 03

 TYPE-IN EXECUTIVE REQUEST:
? usage

 ************************************************
 DASS ACCOUNTING INFORMATION:
 ----------------------------
 START DATE/TIME       ..... : 82/12/30 @ 04.33.16.
 CPU USAGE (SECONDS)   ..... : 15
 EXIT TYPE             ..... : >>NORMAL
 DATABASE CALLS        ..... : 20
 GRAPHICS FACILITY CALLS... : 0
 USAGE CHARGE          ..... : $ 3.00
 FIN DATE/TIME         ..... : 82/12/30 @ 05.14.55.
 ************************************************

 TYPE-IN EXECUTIVE REQUEST:
? time
 TIME ----> 05.15.21.

 TYPE-IN EXECUTIVE REQUEST:
? status
 ECHO=OFF,PLOT=OFF,ACOM=OFF,MDBA=OFF,MGRA=OFF,DBUG=OFF,TEST=OFF
 QUIK=ON ,GROP=0,DBOP=1,ERDF=0

 TYPE-IN EXECUTIVE REQUEST:
? set acom=on
 SET ACOM=ON  COMPLETE...

 TYPE-IN EXECUTIVE REQUEST:
? status
 ECHO=OFF,PLOT=OFF,ACOM=ON ,MDBA=OFF,MGRA=OFF,DBUG=OFF,TEST=OFF
 QUIK=ON ,GROP=0,DBOP=1,ERDF=0

 TYPE-IN EXECUTIVE REQUEST:
? reset
 RESET COMPLETE....

 TYPE-IN EXECUTIVE REQUEST:
? date
 DATE .

 DATE ----> 82/12/30.

 TYPE-IN EXECUTIVE REQUEST:
? cpu
 CPU

 CPU  ----> 15 SECS

 TYPE-IN EXECUTIVE REQUEST:
? $exit
 $EXIT
```

APPENDIX 6

DASS TRAINING

```
$help,t
```



```
        DASS VER.3R1 82.JAN.22

THE DASS TRAINING CONSISTS OF SEVERAL TUTORIAL MODULES. AT THE
END OF EACH MODULE, DASS WILL PAUSE FOR A USER RESPONSE.
THE FOLLOWING USER RESPONSES ARE VALID:

EXIT  - EXIT FROM TRAINING MODE
NEXT  - GO TO THE NEXT MODULE
PREV  - GO TO THE PREVIOUS MODULE
FIRST - GO TO THE FIRST MODULE
LAST  - GO TO THE LAST MODULE
<><><>
 WHEN YOU ARE READY TYPE-IN ---> NEXT
? next

----------
MODULE #01
----------
DASS IS AN INTEGRATED DATA ADMINISTRATION SUPPORT FACILITY.
THE SYSTEM IS MENU DRIVEN - THE USER CALLS 'DASS', AND THEN
SELECTS THE APPROPRIATE MENU-ITEM TO PERFORM THE DESIRED TASK.

DASS SUPPORTS A TEXT/PICTURE ORIENTED DATA DESIGN LANGUAGE FOR
TASK SPECIFICATION & EXECUTION.

DASS PROVIDES A GLOBAL 'HELP' FACILITY - ONCE IN 'HELP' MODE,
THE USER CAN:

 - OBTAIN EXPLANATION ON ERRORMSGS/COMMANDS
 - OBTAIN TRAINING
 - ALTER 'DASS' OPERATING ENVIRONMENT

UPON EXIT FROM THE 'HELP' MODE, THE CONTROL IS PASSED BACK
TO THE 'DETOUR' POINT TO RESUME NORMAL PROCESSING.
<><><>
 WHEN YOU ARE READY TYPE-IN ---> NEXT
?
```

```
next

----------
MODULE #02
----------
DASS DATA DESIGN LANGUAGE OFFERS A VERY FLEXIBLE SYNTAX.

EXAMPLE:
WHILE BUILDING AN ERD, TO DRAW A LINE BETWEEN TWO BLOCKS
#11 AND 12, THE COMMAND IS: 'ADD LINE AT 11,12'.
DASS, HOWEVER WILL CORRECTLY INTERPRET THE COMMAND IF IT IS
CODED IN ANY OF THE FOLLOWING FORMS:
 . ADD LINE AT 11,12
 . ADD LINE @11,12
 . ADD L 11,12
 . ADD LINE BETWEEN BLOCKS 11 AND 12
 . DRAW LINE @11 AND 12
 . CONNECT BLOCKS 11 & 12
 . CONNECT 11,12
 . JOIN 11,12
 . ETC...
<><><>
 WHEN YOU ARE READY TYPE-IN ---> NEXT
? next

----------
MODULE #03
----------
DASS PROVIDES CAPABILITIES TO ALTER THE OPERATING ENVIRONMENT.
THE OPTIONS PROVIDED ARE:

ACOM - ASSUME COMMAND
MCOM - MONITOR COMMAND
MDBA - MONITOR DATABASE ACCESS
MGRA - MONITOR GRAPHICS FACILITY ACCESS
PLOT - ENABLE 'PLOT' CAPABILITY
QUIK - ENABLE 'QUICK' MODE OPERATION
ECHO - ECHO TERMINAL INPUT
DBUG - SET DBUG MODE  - A DASS TRACE FEATURE
TEST - SET TEST MODE  - A DASS TEST FEATURE

THE OPERATIONAL ENVIRONMENT OPTIONS ARE SET USING THE '$SET'
CONTROL COMMAND. DASS CONTROL COMMANDS ARE EXPLAINED IN THE
NEXT MODULE.
<><><>
 WHEN YOU ARE READY TYPE-IN ---> NEXT
? $exit
```

APPENDIX 7

DASS HARDWARE/SOFTWARE ENVIRONMENT

DASS HARDWARE/SOFTWARE ENVIRONMENT

MAIN FRAME

COMPUTER	:	o CDC CYBER 172
OPERATING SYSTEM	:	o NOS (NETWORK OPERATING SYSTEM)
OPERATING SYSTEM VERSION & LEVEL	:	o NOS 1.4, LEVEL 552

SOFTWARE

COMPILER	:	o FORTRAN - FTN5
COMPILER VERSION & LEVEL	:	o FTN 5.1, LEVEL 528
DBMS	:	o GPLAN
DBMS VERSION & LEVEL	:	o GPLAN X.X, LEVEL 123

GRAPHICS FACILITIES

GRAPHICS TERMINAL	:	o TEKTRONIX 4010 o STORAGE-DISPLAY CRT o SCREEN SIZE: 8" by 6" with 1024 x 780 ADDRESSABLE POINTS o INPUT DEVICES:ALPHANUMERIC KEY BOARD, JOYSTICK AND/OR THUMB-WHEEL CONTROLLED CROSS-HAIR CURSOR
GRAPHICS SOFTWARE	:	o TERMINAL CONTROL SYSTEM (TCS),RELEASE 3
HARD COPY UNIT	:	o TEKTRONIX 4631
DISPLAY PLOTTER	:	o TEKTRONIX 4663 INTERACTIVE DISPLAY PLOTTER

DASS CHARACTERISTICS

LANGUAGE USED	:	o FORTRAN
NUMBER OF MODULES	:	o 30
CODE LENGTH	:	o Approx. 6,000 lines