UNIVERSITY OF SOUTHAMPTON

AUTOMATIC GENERATION OF DATABASE SCHEMA

by

L M HOBBS

Thesis submitted in candidature for
the degree of Doctor of Philosophy
at the University of Southampton

1987

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND SCIENCE

ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

AUTOMATIC GENERATION OF DATABASE SCHEMA

by Lilian Mary Hobbs

Designing a computer database is a very complex task. It is a lengthy and time consuming process which involves investigating the business thoroughly, understanding the transactions that will be applied against the database and then translating this into the physical design. One of the major problems with this task is that once the information has been collected, it must then be cross-referenced, collated, documented and then used to formulate the design. When a system comprises hundreds of data items and tens of transactions, this is a very difficult manual task.

What this research shows is that it is possible to develop a computer system which performs all of these tasks. Using the Entity Model as the starting point for the design, the attributes are defined. Then the transactions are described as text and verified, using a natural language processor against the information already entered into the system. Finally the schemas for a CODASYL compliant database are created.

The system is now described in detail, using as an example to illustrate how a database may be created, a simple system for processing policies in a Life Assurance Company.

In memory of my Father

and my brother Reggie

Acknowledgements

I would like to thank John Friend and Schroder Financial Management Ltd, without whose sponsorship this work would have not been possible.

Special thanks to Professor Barron for reading the drafts of this thesis and Mike Freeston for his comments when the system was initially being developed.

Finally a big thank you to my mother for the encouragement that she gave me.

Lilian Hobbs
1987

CONTENTS

CHAPTER 1

INTRODUCTION

From the day that man learnt to write, he has been collecting information. Around 668 - 630 B.C. there was a library at Ninerva that kept approximately 30000 tablets which contained information such as receipts, dictionaries and tables of weights. By 200 B.C. the library in Alexandria contained nearly 700000 rolls of papyrus, but finding data in this library proved to be a problem. So the rolls of papyrus were stored in buckets, each bucket contained a portrait medallion of the writer, which made it considerably easier to locate their work. We have progressed considerably in nearly 2000 years, but these systems were probably the early fore-runners of todays modern database management systems.

A database is defined as a collection of data which is accessed by more than one person and probably used for more than one purpose. Therefore the term database today can be applied to not only data held on a computer but also to any collection of manual records.

This thesis will be concentrating on computer database management systems, where a database management system is defined as a set of resources that control the database by:-

* Storing the data
* Supplying the user with the necessary input/output routines to access the data
* Maintaining security by enforcing privacy and integrity constraints
* Providing backups for recovery from hardware and software faults

Once a database has been established, it becomes an important part of any organisation. By centralizing the data, the following benefits result:-

* Duplication of data is eliminated
* Inconsistencies in the data are minimised
* The data can be shared by several applications
* Standards can be enforced
* Easier to maintain security and enforce privacy
* Standard data formats are then used by all applications

All of the database management systems currently available are based on a data model, which simply represents the information held in the database as it is viewed by the users of the database. One model will not suit all applications, therefore a number of different models are available for different purposes. In database work three types of data model are generally available, the hierarchical, network and relational model.

All of these models are defined using entities, attributes and relationships. An entity is defined as the basic unit about which data can exist e.g. a client and it is described in terms of its attributes. Typical attributes of the client entity would be name, age and date of birth. A relationship is then defined between entities. Possible relationships are one-to-one, one-to-many and many-to-many e.g. One client may have many policies, therefore a one-to-many relationship exists between the entities client and policy.

In the hierarchical model entities are related to other entities in a tree structure. Therefore it is possible to represent relationships of the type one-to-one or one-to-many but not many-to-many.

The relational model is based upon the mathematical theory of relations. Here a relation is a set of n-tuples (e1,e2....en) where entity e1 belongs to an entity set E1. To use this model efficiently a technique known as normalizing must be applied to the relations in order to construct a reasonable relational model.

Finally there is the network model which is very similar to the hierarchical model, except that the type of relationships that may be defined is far more extensive than the hierarchical model, in that it permits entities to be related to any number of entities.

In 1971 the Data Base Task Group (DBTG) of the Conference on Data System Languages (CODASYL) programming language committee published a report which laid down the guidelines and definitions

for a database management system based on the network model. This report included a description of the language that must be used for the definition of the database and this thesis will concentrate on generating a database definition that conforms closely to these standards. [CODASYL]

It should be stressed at this early stage that although the computer industry has been provided with a standard for the database definition, like all standards, suppliers of computer software will often add their own features or not supply all of the features specified in the standards. Therefore on occasions it may be necessary to deviate slightly from the CODASYL standards, but this has only occurred to allow the database definition compiler to generate a database definition that conforms to its own syntax. Where a deviation occurs a suitable comment will be made.

Many organisations have large databases but the construction and on-going maintenance is not a task to be accepted lightly. Usually there is a small team comprising a Database Administrator, a Data Analyst and probably a Data Designer who are responsible for maintaining, tuning and creating the database. The database software contains utilities to edit and compile new or existing database definitions, in addition utilities are usually available to assist tuning of the database but never are any tools available that help in the initial construction of the database.

This lack of facilities became apparent while working as a Data Analyst when the opportunity arose to investigate a major part

of a Life Assurance system, with a view towards creating a database suitable for the Company's business requirements. At this time the company was using a Bureau and had some small systems in-house which all had their own set of files. It was decided that a centralized database was required and this would be maintained using a Database Management System.

The first task to be performed was to construct an entity model that represented the company's business. An Entity Model is always shown graphically, with no computer software available to draw the diagram, the model was drawn many times by hand. From this model the relationships between the entities were identified and it could be seen that either the hierarchical or the network model were suitable for this application. The network model was eventually chosen, because the Database Management Software that had been selected was based upon this model.

Once the database management system had been selected Data Analysis began. This involved collecting information about all of the data currently used by any manual or computer system that would eventually be held on the database. Once the data has been collected, normal practise is then to analyze the data manually, typical tasks include eliminating duplicate data items and removing multiple items. Then one has to analyze all of the transactions that will be applied against the database and see if each one is feasible, if not why not and then how can the model be amended to cater for this transaction type. When there are nearly one thousand data items and well over one hundred transaction types this was an

unenviable task, which proved to be very prone to errors when performed manually and very time consuming. Eventually the database schema which is the actual definition of the database was formulated by hand from this information. At this stage it is very easy to omit a record or data item, or inadvertenly change the definition of a data item. Even entering the definition into the computer took at least one day. This whole exercise took two people many months to complete and then only part of the system had been investigated thoroughly.

It was very apparent during this process that whilst analyzing the business was straightforward, collating, designing, verifying the design and documenting all of the data proved to be tedious and difficult. What was needed was a computer system that enabled one to document the entity model, specify all of the manual forms,report layouts, input screens and data items that will be used in the system. Then define the transactions that will be applied against the database and let the system validate these transactions against the model. Using all of this information the system will then create a CODASYL schema, the storage schema which defines the physical storage of the database and the numerous sub-schema views which are used by the applications to access the database. However, there seemed to be no commercially available system that provided all of these facilities. Using Computer Aided Design Systems one can draw the entity model, thus saving drawing time, but validation of the model is not possible. There are also a number of Data Dictionary systems available that can help during the attribute

definition phase by providing the user with the ability to maintain details about the attributes, records and files used in a system. Unfortunately there was not one system that would take the database designer from the data analysis stage, through to physical database design.

Therefore there seemed to be a need for a system that could perform all of these functions. A system such as this should not be seen as an alternative to employing a database designer. It is merely a tool for the designer to use to assist and quicken the process of development, and hopefully reduce the number of errors that would have occurred if the design was performed manually. During the validation phase a computer system can interrogate the data considerably quicker and with greater accuracy than an individual doing this by hand, thereby reducing the possibility of errors occuring in the design which would then only be discovered when the applications tried to use the database. Errors in the design must be detected as early as possible, otherwise systems development may be delayed for several days or even a week while the application teams are waiting for the error to be corrected. Additional delays will also result if certain parts of the system have to re-tested, to ensure that no new errors have been introduced into the system by the corrections made. It is anticipated that the schemas created by this system will only require minor amendment, if at all, to actually create the database.

Using this system the whole design process is automatically documented and readily available for amendment. Once the database has been designed, the documentation is then available to the project teams to help them during program development. This system will hopefully remove many of the tedious tasks associated with database design and give the Database Designer more time to concentrate their efforts on the more important tasks.

The system being developed will achieve all of these aims and to encourage use of this system considerable emphasis has been placed on making the system user friendly and easy to use. For that reason, all data entry is on-line using formatted input screens, and there is no requirement to learn any special language to use this tool. This makes it possible for the data entry to be performed by VDU operators or any other competent typist. The system has been developed on a VAX computer using the VMS Operating System,TDMS screen handling software and VAX11-DBMS Database Management System for the database. The programs have been written in DELTA, which is a fourth generation language that creates ANSI COBOL. [DELTA] DELTA has its own language but the great advantage is that only a few lines of code need to be written to generate the COBOL equivalent. Hence program development is quicker and the application code is much easier to read. One does also not need to be a COBOL programmer to use DELTA, a knowledge of DELTA is all that is required.

Although this effectively means that the system can only be used on a DEC VAX computer [VAX], because of the modular approach of the DELTA language, provided the on-line interfaces are amended. The system can be amended to run on any computer that has a CODASYL compliant database and a DELTA generator. DELTA compilers are currently available for IBM , ICL , DEC and PRIME machines. Only the software and hardware available in the Commercial environment in which the system was developed was used to develop this system.

Since a complete system such as this has never been attempted before, it was felt that an essential part of the thesis was the actual creation of a prototype system. Rather than merely describe the features that could be included in such a system. The majority of this thesis is a detailed narrative of the prototype system.

CHAPTER 2

DATABASE DESIGN TOOLS THAT ARE CURRENTLY AVAILABLE

2.1 INTRODUCTION

A vast amount has been written about how to design a database, and there are various recommended methodologies, but there are no software tools available which take the database designer from the data analysis stage, through to physical database design. Some software tools are available, but they are limited in the facilities they provide, frequently they only run on certain machines, are very expensive and none of them will actually create a CODASYL schema.

Unlike other software products, very little publicity seems to be given to Database Design Tools. Perhaps this is due to the fact that they have a limited market and are often difficult for a commercial organization to cost-justify, because the benefits they provide are often intangible. Therefore the first major problem was trying to establish what software products were available.

The International Directory of Software [INTDIR 81] identified two types of products that could be used during Database Design. They are a Data Dictionary System and Design and Documentation Tools. First we will look at the Data Dictionary systems that are available as these are quite numerous across the various machines, then we will review the Design and Documentation Tools that are available.

2.2 DATA DICTIONARY

A Data Dictionary System is defined as any system which holds information about the data that an organization uses and processes. Initially a Data Dictionary would seem to be the ideal tool to use during database design, because it holds details about all of the data and processes that a company performs, it should be possible to create a schema from the information contained within the Dictionary. Remember, a system that allowed definition of the entity model, attributes, transactions and then created the schemas, was the requirement. However, as each system was examined they all seemed to fall very short of this requirement, in fact it was surprising to discover just how limited some of the packages were. [BCS83] [BCSWRP] [LOMAX] [MAYNE]

The following Data Dictionaries were assessed for their suitability:-

* ADABAS
* Central Data Dictionary
* Data Catalogue 2
* DDS
* GARDE
* IBM DB/DC
* IDD

2.2.1 ADABAS

ADABAS, an acronym for Adapatable Database Management System, is a database management system which supports either the relational or network model, and it comes complete with a Data Dictionary System. Although the dictionary facility is only really of use to ADABAS database users, before disregarding it, it is worth highlighting some of the facilities that are available within this system.

Using a menu the user selects the task to be performed and a nicely designed formatted screen is provided for data entry. One can define a file by specifying its name, type and size. Detailed information about the attributes of the file can then be defined such as format, short name, length etc. Other facilities include the ability to define a report name and specify the files from which

the data for the report will be derived. Brief program details are also kept which include program name, author, language used, comments and modules or reports. The modules within a program may also be defined in a similar manner.

This system provides a place where information can be stored once the design decisions have been made. It is a very user-friendly system and the on-line interfaces make interrogation of the data very easy. Unfortunately it is totally unsuitable for this application because it does not have the ability to store or create a CODASYL database definition. [MAYNE]

2.2.2 Central Data Dictionary

This is a package marketed by the Digital Equipment Corporation for their VAX range of computers, in a recent survey it was named as the third most popular data dictionary system. [COMPDD] [VAX] Using the Data Description Language, record and database definitions only may be inserted into the dictionary, there is no facility to define transaction details or the entity model. A standard text editor has to be used to create the record definition which is then compiled into the dictionary. To define a record comprising of data items Policy Number , an eight character field, the Policy Start Date which is packed decimal, a typical record entry is shown in figure 2.1

```
DEFINE RECORD POLICY_REC.
  POLICY STRUCTURE.
   /* Policy Number */.
    POLICY_NO           DATATYPE IS TEXT
                        SIZE IS 8 CHARACTERS.
   /* Policy Start Date */.
    POL_STRT_DT         DATATYPE IS PACKED DECIMAL
                        SIZE IS 8 DIGITS.
  END POLICY STRUCTURE.
END POLICY_REC RECORD.
```

Figure 2.1

Apart from listing the definition no other reports are available. It is not possible for one to compile a database definition into the dictionary directly, instead the database management software automatically places the definitions here when the request to compile a definition is initiated.

This software merely provides a place where definitions may be held centrally once all of the design decisions have been made manually. Of all the Data Dictionary systems reviewed, this one is probably the most limited in the facilities provided. [CDDDEC]

2.2.3 Data Catalogue 2

This system is available for IBM, Honeywell and Sperry Univac machines. It is a comprehensive system which allows one to specify individual items, records, files, computer processes, forms, report layouts, external resources, manual tasks and the users responsible. Relationships may be defined between procedures, and between procedures, users and data, but one cannot actually define the entity model.

The method of data entry is flexible. Using a data definition language, data may be entered either batch or on-line in either a free-format, tutorial prompt or via tailored forms for batch entry. In free form mode typical statements are shown in figure 2.2 which illustrates how the new attribute policy number may be defined.

```
ADD ELEMENT = POLICY-NO
DESC
1   A Unique reference assigned to a Policy
10  The numbers are allocated in sequence
15  and there is no check digit.
```

Figure 2.2

A comprehensive set of reports is available and a query language is available to interrogate the data. However, the documentation produced is not readily understood, because the numbers allocated to each line of description are particularly confusing to the first-time reader when the free-format mode is selected.

A recent enhancement to the system is the Custom Dictionary Facility (FACETS). It provides the user with the ability to define information about the organization, a procedure, function, source of data etc, which then enables the data model to be defined. The system will then generate a logical view of the database , which is merely a model of the information defined previously. The physical schema still has to be created manually, using this logical view as the basis for the translation.

2.2.4 DDS, The ICL Data Dictionary System

This is a very comprehensive system for ICL machines and one of the few systems that actually allows the user to specify the Entity Model. Detailed information about individual attributes is also retained, these attributes may then be grouped and formed into records and files. One can also specify the systems and processes that create or use the entities and the events that result or are caused by entities. The major drawback with this system is that all of this information is defined using a very rigid syntax, which may be input in either batch or on-line. Therefore the system is not very user-friendly, especially as there are no formatted screens available to assist during data entry. A typical record entry as per the example for the Central Data Dictionary (see fig 2.1) is shown in figure 2.3. Whilst this syntax is familar to COBOL programmers, there are many people who would find this format very difficult to understand, especially analysts who have never been COBOL programmers.

```
INSERT RECORD POLICY_REC
* STRUCTURE
    ITEM  POLICY_NO
    ITEM  POL_STRT_DT
**
ITEM POLICY_NO
* DESCRIPTION Policy Number
* PICTURE X(8)
ITEM POL_STRT_DT
* DESCRIPTION Policy Start Date
* PIC S9(8)
* USAGE COMP-3
```

Figure 2.3

Using the rigid syntax a CODASYL schema may be defined in terms of Areas, Records and Sets and from this information a schema for the IDMS database management system can be generated. Various reports are also available in respect of the data in the Dictionary, which may be interrogated via the on-line interfaces, however the lack of formatted screens does not make selection of data easy.

This system almost satisifies the original criteria because it allows definition of the entity model,attributes and the transactions. Where it is limiting is because the system can only validate something if it knows what it is. Therefore during the transaction phase it has to be told that the following items are records or attributes, it is unable to decide for itself what something is. In addition this system will only create the Schema provided the area,record and set definitions have already been supplied. That means the designer must construct the physical schema manually, and once the decisions have been made as to the areas,records and sets that are required. This information has to be manually entered into the system so that the physical schema can be created. Hence the system is providing little more than a text-editting facility for the definition of the database and it is not actually deriving the schema from the information in the dictionary. Finally there is only limited cross-checking of data in the dictionary when it is being input. This is an ideal tool for retaining information collected during data analysis and design. [BCS 83] [ICLDDS] [LOMAX] [MAYNE]

2.2.5 GARDE

A system developed in France for IBM machines, it is designed primarily for the hierarchical based Database Management System DL/1 that runs on this equipment. The system is worthy of mention because it provides the user with a very friendly online enquiry/update facility. Using well designed screens information may be entered about the attributes, and text supplied regarding the transactions , although this is not validated. The database definitions may be supplied, but there is no facility to enter the entity model. Once these details have been defined the system will generate the DL/1 control blocks that are needed to define the database. Once again this is a very useful system for documenting the database design and the only system where the screen definitions are in French. [GARDE]

2.2.6 IBM DB/DC

This system is designed specifically for IBM machines and for holding details about the IBM hierarchical database management system. The data may be entered on-line or in batch using a free form command language, or via a comprehensive set of 'Fill in the blanks' on-line dialogue. Information is retained about elements, records and data sets only. The dictionary also offers a facility known at the 'Extension' option, where new categories and relationships between categories may be defined. The dictionary is installed with the basic categories such as elements and segments,

where an element is an attribute and a segment is a record. Using this facility it is possible to extend the dictionary to define the entity model and transaction details by defining these as new category types. A typical definition of a relationship is shown in figure 2.4 where a relationship is being defined between the policy and client entity.

```
ADD ENTITY POLICY
ADD ENTITY CLIENT
ADD_RELATIONSHIP ENTITY POLICY WITH ENTITY CLIENT
```

Figure 2.4

The dictionary is very comprehensive and the extension options create a very powerful tool. Yet once again it lacks the ability to actually create the schema. It will only create it provided the actual details of the Schema have been input. [BCS83] [LOMAX] [MAYNE]

2.2.7 Cullinane Integrated Data Dictionary (IDD)

The IDD is designed to support the documentation of all data and its utilisation. It runs on IBM machines and it is a comprehensive package that enables the user to specify individual items, records, files, modules, reports, entry points, users, processes and schemas for an IDMS database. The IDMS database management system is Cullinane's own system that is based on the network model.

The information is defined using the Data Dictionary Definition Language compiler, refer Figure 2.5, which can be used in either batch or on-line mode. As can be seen from the example in figure 2.5, which is defining the process of Direct Debit Collection, the dialogue is not very user-friendly.

```
ADD PROGRAM DDCOLLECTION 0001
USER IS ACCOUNT_DEPT
INPUT FILE IS DD_DETAILS
OUTPUT FILE IS DD_COLL
PROGRAM IS CALLED DDCOLL
LANGUAGE IS COBOL
```

Figure 2.5

The system comes complete with a full range of reports and supports on-line query of the data.

At the moment it is not possible to define the entity model and processes may only be described using this rigid syntax, rather than in general business terms. In a future release it is hoped to provide the ability to describe the business descriptions of the data and its processes. These will then be used to create a fully designed database schema and subschemas, but no information is currently available as to when this extension will be released.

Like the ICL Data Dictionary System, IDD is an ideal system for retaining design decisions once they have been made, with over 700 user worldwide it is a very popular package. [BCS 83] [LOMAX] [MAYNE]

2.2.8 Other Dictionaries

A number of other Data Dictionary systems were assessed, they included BYBLOS, Honeywell's DD/DS, Cincom, UCC 10 and Saxiom. All of these were for specific machines and performed basically the same functions. That is, one can store details about attributes, records, files and databases using a data description language. These definitions can then be extracted into an application or used to produce reports about the data in the dictionary.[MAYNE]

All of the tools mentioned so far are merely Data Dictionary systems, that are very good at documenting the data collected during data analysis and design. They all allow the user to specify record or database layouts and then extract this information into a program or some other machine readable form, but unfortunately that is as far as they go.

2.2.9 Design And Documentation Tools

The next set of tools to be reviewed are Design and Documentation Tools or Data Dictionary systems that include an additional interface to a data design tool.

The following were reviewed:-

* Data Designer
* Designmanager
* Excelerator
* Information Engineering Workbench

* LBMS Auto-Mate
* LBMS Data-Mate
* PRIDE - ASDM - Automated Systems Design Methodology

It was very interesting to see the new tools that were reviewed in a more recent edition of the International Directory of Software [INTDIR 84]. Compared with the 1981 edition, the trend in the Documentation tools sections seemed to indicate that companies were now concentrating their efforts on developing cross-reference and charting tools, apart from LBMS' Auto-Mate and Data-Mate products, there was not one new Database Design Tool amongst the new entries.

2.2.10 Data Designer

This is an interactive or batch design tool for VAX and IBM machines which is marketed as an aid to the database administrator for creating logical database designs.

To use the tool, the data analyst must first formulate user views of the required accesses to be made upon the database. It is recommended that these views are initially represented as bubble charts, where a bubble chart is a graphical representation of a user view that can be easily understood by both users and systems personnel. Every item is drawn with an oval shape around it and different types of lines are drawn between the items depending on the associations with the data, hence the term a bubble chart.

[MARTIN]

Using a rigid syntax the user specifies the data items to be included in the dictionary and a typical dialogue is shown in figure 2.6, where the items policy number and policy start date have been placed in the dictionary.

```
>BUILD  POLICY.DIC          (Create a dictionary file
 FILE TYPE: $DIC             called Policy)

B> POLICY-NO
B> POLICY-STRT-DT
```

Figure 2.6

When all of the items have been defined in this way, note that there is no facility to enter detailed information about the items such as picture formats and description. The user view of the data is input into a subschema file. A typical user view entry, the report Policy Report is shown in figure 2.7

```
>BUILD   POLICY.SUB            (Create Policy Subschema
 FILE TYPE: $SUB                 File)

B> V,POLICY-REPORT             (V denotes a User View)
B> F,1000                      (Frequency)
B> T,1                         (Response Time)
B> K,POLICY-NO                 (Key)
B> 1,POLICY-STRT-DT            (1-1 relationship with key)
B> M,PAYEE                     (1-M relationship with key)
```

Figure 2.7

These subschema files are then validated against the dictionary, by checking for data items in the user views that are not in the main dictionary. Various reports are also available and one that is particularly worthy of mention is the calculating and reporting of response times for a given item, dependant on the volume of of data being accessed. Once the validation is complete the system will generate the logical design in either a graphical or report format, which is then manually translated to a CODASYL schema.

Unlike the other systems reviewed this software is not a Data Dictionary system, but a true database design tool. By supplying the system with only limited information it will formulate the design for a number of database management systems, from which the designer can translate this into the physical design. One limitation of the system is that it does not allow the information collected during data analysis to be defined within this system. Therefore a Data Dictionary is still required to record all of the information collected, and there needs to be some mechanism for transferring the information from this system to the Data Dictionary. This is a good tool for the designer to use while trying to formulate the database design. [DATADES]

2.2.11 DATAMANAGER And DESIGNMANAGER

DATAMANAGER is a very popular data dictionary system for IBM machines which boasts a range of additional modules which enable the

system to be used on-line and to interface to various DBMS'. DATAMANAGER is the data dictionary module that allows the user to define files, records, groups, attributes and modules using a rigid syntax.

An additional module that may be purchased is DESIGNMANAGER. It supports the collection of data using either the 'Top-Down' (specify the entities and their associations and then supply the detail later) or the 'Bottom-Up' technique (specify the data items and the dependencies later). This information is loaded into a separate area known as the Workbench Design Area which may also comprise of data that has been down-loaded from the DATAMANAGER dictionary. All of this information is then formed into the composite view, then by normalizing these views the conceptual schema is generated. The system will graphically display the Conceptual Schema and generate reports showing each record with its items and all of the records that either leave or point to it. It will also model the physical access times for data elements such as minimum/maximum response times. This conceptual schema can then easily be converted to a Relational DBMS. Alternatively if the DBMS is DB2, SQL/DS (both relational systems) or IMS (hierarchical system) then the system will actually generate the First Cut Database Design.

This software provides the facility to formulate a design in a separate area, but because there is no requirement to keep the information in DATAMANAGER and DESIGNMANAGER identical, differences between the two modules could occur. This is the only system

reviewed that made any attempt at trying to create the first cut database design. Unfortunately for a CODASYL database the generation of the Schema still has to be performed manually from the Conceptual Schema. [BCS 83] [DSGNMGR] [MAYNE]

2.2.12 Excelerator

A new product launched in August 1984, it is designed to run on IBM PC's and compatibles. Using the extensive graphics capability one can construct Data Flow and Data Model Diagrams according to the methodology selected, and document record layouts, report and screen formats. The system can then be interfaced to a word processing system to generate quality documentation.

This tool is only a documentation aid, it generates quality documentation and it is ideal for maintaining the entity model or final database diagram, but it has no facility to use the information entered to generate a Schema of any kind.

2.2.13 Information Engineering Workbench

This is another recently launched product for the IBM Personal Computer from KnowledgeWare Incorporated. Even in its initial release it is an impressive tool that either the Systems Analyst or Data Analyst can use.

Using computer aided design techniques the user may specify data flow diagrams and the entity model. A facility called the Decomposition Diagrammer enables further details to be specified to the diagram by adding refinements to the diagrams and any modules that have previously been defined. The Entity model can be defined using either standard rectangles or diamonds and this is the only system reviewed that allowed one to specify whether a relationship between entities is optional or mandatory. Detailed information about the attributes may be defined, along with references to any data flow diagrams where this entity may be referenced. Action diagrams may be constructed which illustrate the logic of an activity using both graphics and text. The information being entered is constantly being validated against details already held in the Encyclopedia, where the Encyclopedia is the name given to the place where all the data entered is retained. An expert system performs the role of checking all the data for consistency and corrections and translating text into diagrams and vice versa.

Eventually this software will interface to other design products one of which is Data Designer. At the moment it generates high quality documentation and provides extensive data cross-checking. Since it can be used by both the Systems Analyst and the Data Analyst, everyone can share the same information base and investigations need not be repeated by several people. The current release does not make any attempt at creating a schema either the conceptual or physical one. Therefore at the moment this tool is probably of more use to the System Analyst rather than the

Data Analyst. [IEW]

2.2.14 LBMS AUTO-MATE

At the end of 1986 a new product was launched by Learmonth and Burchett Management System called Auto-Mate. It has been designed to support the LBMS Structured Development Method [HALL] and it supports all of the steps of logical specification. The new generation of software tools are starting to be developed for Personal Computers, and this product is no exception. Developed for the IBM PC, the system makes extensive use of the latest graphics facilities. This enables the user to define Data Flow Diagrams and Logical Data Structures using the symbols associated with Structured Systems Analysis Design Methodology (SSADM).

Using well designed screens, this user-friendly and easy to use system allows the user to define the entities and their relationships, attributes, processes, functions and event descriptions. However, the system does not try to interpolate what has been entered, it merely checks that the definitions exists in one of the Data Flow Diagrams or Logical Data Structures previously defined. All of this information is then reported in the form of high quality documentation. In its current state this is a perfect tool for the System Analyst to use when analyzing and documenting a computer system.

2.2.15 LBMS DATA-MATE

Launched in 1986 this is a product aimed specifically at the Data Analyst, which integrates fully to the previous mentioned product Auto-Mate. Using the graphics capability of the IBM PC, the entity model is defined and then detailed information is supplied about each entity. Next the attributes can be defined and if required the system will attempt to perform normalization down to the third normal form. A very unique output from this stage is a diagram of the results of this step.

Running on an IBM PC the system has the disadvantage that even if it could create the schema definitions, it cannot directly interface to the machine where the database would reside. There is no attempt at first cut database design, but nevertheless the procedure for defining the entity model is one of the best available. The inability to define the processing information may be a disadvantage to some users. The product is probably aimed at the relational database user rather than large CODASYL databases because there has recently been a tremendous growth in the relational database market. With a lack of suitably qualified database designers this system will enable any competent analyst to have a reasonable attempt at database design.

2.2.16 PRIDE - ASDM- Automated System Design Methodology

This provides a totally integrated and automated approach for the specification, analysis, design, development and

implementation of Information systems.

The Automatic Design Facility (ADF) automatically designs and documents information systems and logical design. The Information Resource Manager (IRM) is used to control, catalog and cross-reference all of the data and systems used. There are optional packages available to interface the software to a DBMS, but once again the system will not create the schema, only a basis from which the physical implementation can be formulated. Unlike most of the other systems this software will run on a variety of machines which include IBM, DEC, CDC Honeywell, Univac, DG, Burroughs and Hewlett Packard.

2.3 OTHER DESIGN TOOLS IN DEVELOPMENT

Although several database design tools are available, the subject is still worthy of research because the systems available are incomplete. There is not one that the designer can use from data analysis through to database design. Consequently the subject has received considerable discussion in various papers and via research. [HUBBARD] [MTDBO]

A project was underway in one government installation that involved the creation of a Schema Design Generator for an IDMS network based database which is CODASYL compliant. The system was completely specified and the coding up to the conceptual schema stage was completed when funding for the project ceased. At this time it is not known if work on this project will ever start again.

Another team at the Milan Polytechnic is developing A Natural Language Reasoning System for the Analysis of Database requirements. This is the first system reviewed that allows the user to specify in a natural language the data, transactions, events and constraints. Currently the system is being developed to automate the collection and analysis of database user requirements. Ultimately, but not for some time to come, the system will be enhanced to create the conceptual, logical and physical schema. [COLOMB]

Also being developed at the Milan Polytechnic in the Electronics Department is another system for the conceptual design of database schemas and transactions. This system has been developed on a PDP 11 computer and is written in PASCAL. It is based on the Canonical Binary Model and requires the use of a Schema Definition and a Transaction Definition Langauage, neither of which seem very easy to use. [BRACCHI]

Another two other tools that are under development in Italy which are both being used to create systems for the Conceptual phase of Database Design. One is a system that runs on Honeywell machines where the data is entered using a rigid syntax. [ALTZENI] The other system has been developed using a dialect of the LISP language, MAGMA-LISP, running on a VAX computer under UNIX. [ALBANO]

2.4 ASSESSMENT OF TOOLS CURRENTLY AVAILABLE

Of all the systems reviewed only parts of each system came close to the objectives that had been specified. One of the main problems with these systems is that nearly all relied upon the user using some form of Data Definition Language to input data into the dictionary. Only ADABAS , GARDE , AUTO-MATE and DATA-MATE made any attempt at creating a user-friendly environment. When entering data there also seemed to be very limited cross-checking between data already in the dictionary. A few systems would allow the entity model to be specified and LBMS Auto-Mate was an impressive tool for this purpose. All of the systems were very good at defining the files, records and their attributes, and some systems did allow the transaction details to be defined. But none allowed the transaction to be defined in text form and then try to determine what the transaction was doing and validate this against the information already held in the dictionary.

Once all of the data had been entered, some systems attempted to construct the conceptual schema, but none even contemplated the physical schema for a CODASYL database system. Instead most relied upon the designer manually creating the schema and then using these systems as a safe place to keep the design.

It was worrying to see that the new tools being launched were for personal computers. These machines generally have good graphics capability and storage facilities, but the main concern is the errors that can result when the information is transferred to

the main computer. None of the systems seemed to have addressed this problem and this simple process could ruin nearly all of the benefits that have accrued from using these systems. By placing the information on a Personal Computer the information is also not readily available to the project teams during development. It is pointless retaining all this information if it cannot be shared with the very people who need to use it.

If component parts of each of the systems reviewed were used then a system close to the original requirements could be formulated. What has not been tackled by any system is the use of a natural language processor to accept transaction details in text form and derive from this the processes and actions. Finally the actual generation of the first cut CODASYL schema, storage schema and subschemas has yet to be attempted by any of the systems reviewed. Reviewing all these systems certainly indicates that there is a need for a tool that will provide all of the facilities initially specified.

CHAPTER 3

SCHRODER DATABASE DESIGN SYSTEM

3.1 PRELIMINARIES

From the outset it was decided that unlike other theses which merely theorise, this computer system would actually be created and be put to use in the commercial environment that sponsored its development. The first release of this system will not be as powerful as some of the systems reviewed. However, what will be shown is that the idea of actually creating a CODASYL database is feasible, and that although certain restrictions have been imposed on the first release, this has not greatly affected the performance of the system. It is envisaged that these restrictions will be removed in future releases.

The review of the systems that were currently available revealed a total lack of user-friendliness, therefore constantly during the design process, designing the system so that it was easy to use was of great importance. Entering vast quantities of data into the system is a labo rious task. If this can be performed by a

non-computer person, then the database designers time can be better utilised.

The system had to be developed on a VAX computer running the VMS Operating System and using only the software that was available within the Commercial organization sponsoring the research. The facilities used are typical of those normally available in any Commercial installation. That is standard languages such as COBOL were used and conventional terminals instead of graphic ones were utilised. An organization is often more willing to experiment with new software if it can be run on existing hardware and this restriction does not appear to have greatly reduced the effectiveness of the system. Obviously the system will work best on the VAX hardware range but it could be converted to run on any machine that has COBOL and a CODASYL compliant database, although some additional work will be required to amend the on-line interfaces.

The system itself is written in DELTA, COBOL and BASIC. DELTA is a fourth generation language, which is very modular and provides a fast and efficient means for a non-COBOL programmer to develop COBOL applications. [DELTA] To create a program the programmer uses DELTA and COBOL instructions which are then processed by a DELTA compiler that creates an ANSI COBOL program. This COBOL program is then compiled and linked in the normal manner. By generating ANSI COBOL it is then possible to transfer this software to other machines. Portability of computer software is always very difficult, but any steps that the programmer can take to

make the transfer easier are always well worth making. One of the main problems with most of the software that was reviewed was that they were all very machine specific.

A number of subroutines have been written in BASIC because this language is very good at string manipulation. The BASIC subroutines have only been used in a few cases where it would have been very difficult to write the same code in COBOL. The subroutine is compiled as a separate program and called as a subprogram from the COBOL program.

Choice for the data structures was between using standard files or a network based Database Management System. The VAX-11 DBMS which is CODASYL compliant was chosen, because it offered data integrity, data independence, recovery and journalling facilities to ensure the dictionary did not become corrrupt. It was also felt that the relationships that existed between the data structures could more easily be represented in a network database.

Three screen handling facilities were available, they were MAS-M, DELTA SCREEN or VAX Terminal Data Management System, hereafter referred to as TDMS. MAS-M is a system developed by Hoskyns which was not selected because the method of communicating with the user was not very user-friendly in our installation. The DELTA langauage has its own screen handling system but this required some in-house software to be written to use this facility, which did not make the software very portable. The final option was to use the DEC screen handling facility TDMS, which is a powerful tool that

ensured that the package could be run on any VAX computer.

The programs are mainly run on-line using formatted data entry screens so that the user does not have to worry about learning a special language to use the system. All of the reporting programs have the ability to run either in batch or on-line. However, several of the programs at the end of the system involved in analyzing transaction details and generating schemas are run in batch because it is felt that the response would be too slow to an on-line user.

3.1.1 Screen Handling Features Of TDMS

The screen handling facilities available in TDMS are typical of those available in this type of system. The screen definitions are defined using a special utility and are kept in a library separate to the application. This means that the screens can be amended without the need to re-compile the application program. When the screens are defined validation checks and help information may be specified for each field.

The keyboard keypad is used by the operator to advise the application of the required action. Above the keypad are four keys referred to as PF1 PF2 PF3 and PF4. If PF1 (hereafter referred to as GOLD) is pressed along with the appropriate alphabetic character, the selected action is taken. e.g. If the display line shows 'I - Insert' then pressing GOLD I will instruct the system that this record is to be inserted.

The following standards have been adopted throughout the system:-

A - Amend

C - Complete Data Entry and Continue Option

D - Delete

E - Exit Program

I - Insert

P - Print

Q - Quit current option

S - Display record details

The twenty fourth line of the display is used to send messages to the user, such as what to enter or why a validation check has not succeeded. The system also has a built-in HELP facility. Whenever PF2 is pressed once, a message is displayed on the 24th line indicating what the system expects the user to enter in the current input field, if PF2 is pressed twice then global help is provided explaining what the program actually does. Movement from one field to another is achieved by using Tab, Backspace or the cursor arrow keys and the screen is transmitted by pressing the RETURN key. The system also provides what is known as a 'Scrolled' area, for information that cannot fit all on one screen. Data is entered and when the bottom line of the screen is reached the input area scrolls up one line to allow more data to be entered. This area can be scrolled either up or down using the cursor control keys, it is used to either input or display data. A message to use the cursor control keys is always displayed if a scrolled area is being used. The screen can be re-built at anytime by pressing the

keys Control and W, which is very useful if any system messages disturb the screen format.

3.2 DATA DICTIONARY

No suitable Data Dictionary was available to store all of the information that this system will collect. Therefore a special Data Dictionary has been designed specifically around the needs of this system. Rather than confuse the reader at this point, the usage of each record will be defined at the point in the system where it is used. Figures 3.1 and 3.2 illustrate the structure of the data dictionary.

Figure 3.1 shows the control files used in the system and where the transaction details and schemas created by the system will be held. Whilst figure 3.2 illustrates the complex data structures that have been developed to hold all of the information that the system requires. The attribute is selected as the central item, from here to the right a relationship is shown been the attribute and the entities. Then to the left the relationship with the form and form-type records shows where this attribute is used. Underneath the attribute is the specific information about it i.e. its computer names and synoynms.

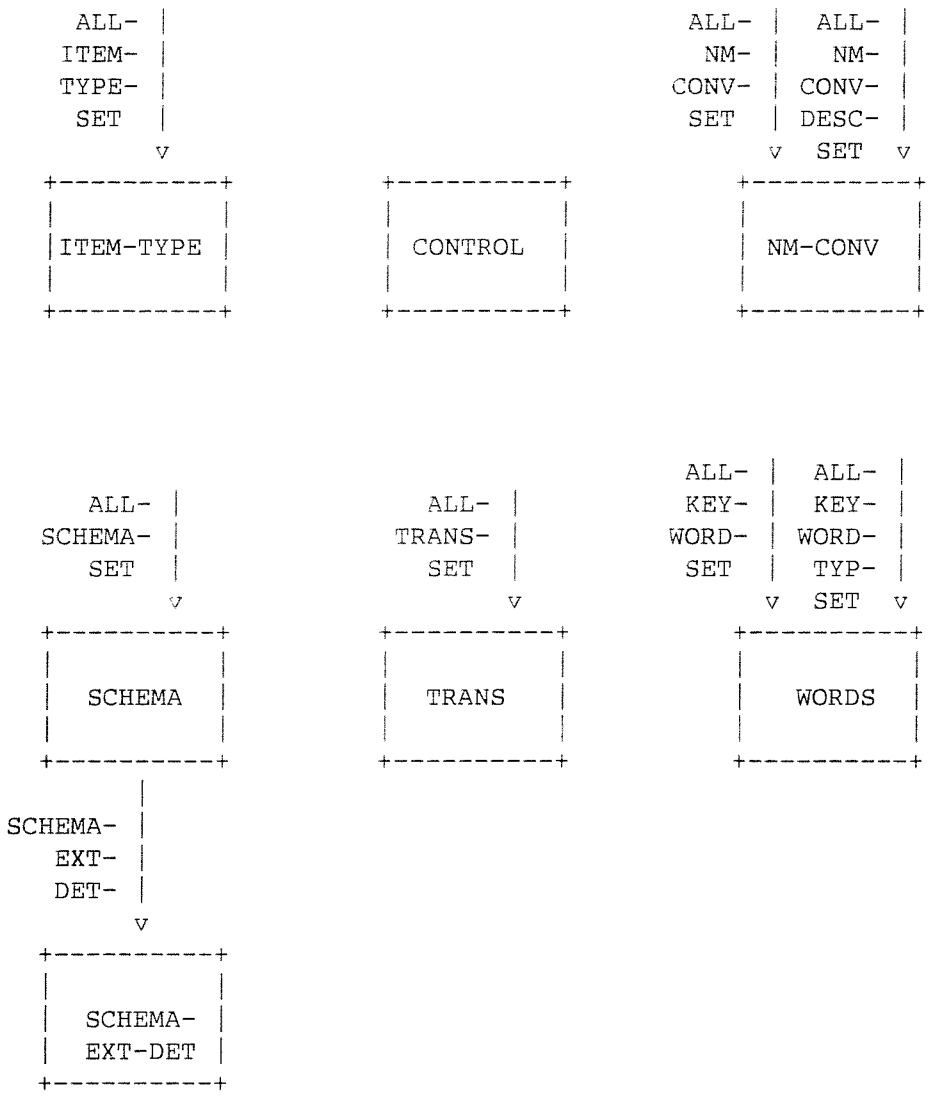

ALL-
ITEM-
TYPE-
SET
ITEM-TYPE
CONTROL
ALL-
NM-
CONV-
SET
ALL-
NM-
CONV-
DESC-
SET
NM-CONV
ALL-
SCHEMA-
SET
SCHEMA
ALL-
TRANS-
SET
TRANS
ALL-
KEY-
WORD-
SET
ALL-
KEY-
WORD-
TYP-
SET
WORDS
SCHEMA-
EXT-
DET-
SCHEMA-
EXT-DET

Figure 3.1

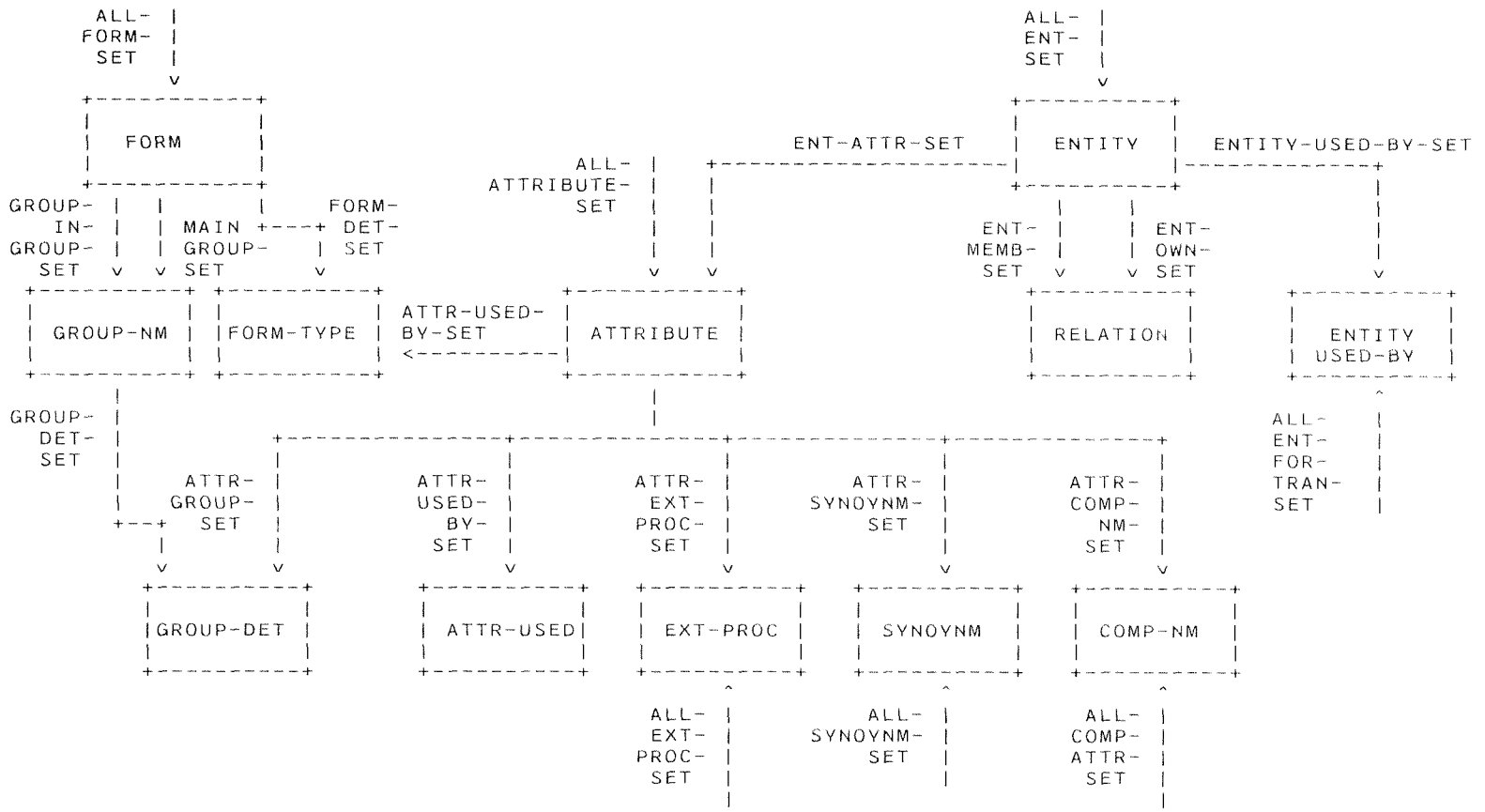

ALL-
FORM-
SET
FORM
GROUP-
IN-
GROUP-
SET
MAIN
GROUP-
SET
FORM-
DET-
SET
GROUP-NM
FORM-TYPE
GROUP-
DET-
SET
ATTR-
GROUP-
SET
GROUP-DET
ATTR-USED-
BY-SET
ATTR-
USED-
BY-
SET
ATTR-USED
ALL-
ATTRIBUTE-
SET
ATTRIBUTE
ATTR-
EXT-
PROC-
SET
EXT-PROC
ALL-
EXT-
PROC-
SET
ENT-ATTR-SET
ATTR-
SYNOYNM-
SET
SYNOYNM
ALL-
SYNOYNM-
SET
ALL-
ENT-
SET
ENTITY
ENT-
MEMB-
SET
ENT-
OWN-
SET
RELATION
ATTR-
COMP-
NM-
SET
COMP-NM
ALL-
COMP-
ATTR-
SET
ENTITY-USED-BY-SET
ENTITY
USED-BY
ALL-
ENT-
FOR-
TRAN-
SET

Figure 3.2

3.3 METHODOLOGY

When it comes to designing a computer system there are various recommended methodologies such as ACM/PCM (Active and Passive Component Modelling), DDSS (Development of Data Sharing Systems), LBMS-SDM (LBMS Systems Development Methodology) which was enhanced to SSADM (Structured Systems Analysis and Design Methodology) and the Jackson System Development methodology to name but a few. [ISM] This system has not been designed around one methodology, because these methodologies are primarily intended for the design of a computer system, of which the database forms a small part. Instead this system has been targeted specifically at the Data Analyst and Database Designer. Since most of the previous mentioned methodologies use data analysis techniques, it is these facilities that this system provides. (Data Analysis is the process of analysing and documenting the data associated with an area of the business and Database Design is the design of the physical database).

Many of the previously mentioned methodologies such as ACM/PCM, DDSS and LBMS-SDM all consider Entity Modelling to be an essential part of capturing and representing data from the real world. Later it will be explained why the Entity Model was chosen, but it is at this point that the system begins. The first group of

facilities allow the definition and maintenance of the Entity Model.

Once the Entity Model has been defined the next step in Data Analysis is to define the attributes of the entities. One technique for finding this information is to examine all the input forms, reports, existing files etc that will be used in the system. From all of these different types of documents the attributes are extracted and allocated to an entity.

What this system enables the user to do is to specify a document i.e. a manual input form or a report and then list brief details about each attribute on the document. The system will then automatically try to allocate this attribute to an entity or the user may specify the entity.

All of the data entered is retained in the Data Dictionary and facilities typical of those to be found in a Data Dictionary system are then available to enquire upon this information. Most of the methodologies mentioned need a data dictionary to retain the information collected during Data Analysis, therefore the system still has appeal to a wide band of users, even though the methods of data collection may vary.

The next stage is to establish which transactions will be applied against the database. The purpose of defining the transactions is to verify that the design being proposed can handle all of the processes that will applied against it. It is at this point that the majority of the systems reviewed seemed to falter. Whilst they allowed the user to record the transaction in the

dictionary, invariably using some rigid syntax, that was the extent of the assistance that they provided. Consequently, the validation of the transaction against the model had to be performed manually.

It is at this point that this system differs from its counterparts. What it allows the user to do is to specify the transactions in text or any required format and when requested the transaction will be validated against the model. This is achieved by developing an elementary natural language system. Using the information already in the dictionary, the system looks at each word or groups of words and then tries to identify then as entities, types of documents, attributes or English words. A facility is available to define extra words and their type which enables the system to understand what the transaction is doing. Examples of extra English words are, 'the' or 'a', and process words such as create and delete. It will also check that access paths are available and that suitable entry points have been defined.

The result of this verification is the generation of a very simple diagram that illustrates the flow of the transaction through the proposed model and a list of words that the system didnot understand. By a process of iteration and modification, the model can be amended to process the transaction. Since each transaction is carefully analyzed, the system is more likely to detect any inconsistencies with what has been previously defined than an individual performing the task manually.

How effectively this part of the system performs is dependent upon the data that has previously been defined. In its initial release, this part of the system may have some difficulty analyzing very complex transaction. But, what it does show is that the validation of the transaction is feasible and this area will be enhanced in future releases.

The system will then if required, perform first normal form normalisation on the data. Normalisation is optional because it is a technique that is not always required, because in some organizations certain duplication of data and multiple occurrences of items is desirable.

Finally the system will attempt to create the CODASYL schema and storage schema, which is a feature that none of the systems reviewed attempted. Then the required subschemas can also be created by specifying which transactions will be processed, and then using the information specified at the transaction definition phase the subschema will be generated.

The purpose of this system is to reduce the number of errors that can occur during design, too many database systems fail due to errors that have occurred in the original design. As the data is entered into the system, it is continually being validated, therefore the chances of errors occurring is greatly reduced. When this is performed manually it is too easy to lose a piece of paper or accidentally overlook some important aspect of the system. It can be seen that this system performs all of the basic functions

that are normally required when developing a database, without restricting the user to a specific methodology. This system should never be seen as an alternative to employing a Database Designer, it is merely a tool for this person to use.

3.4 SYSTEM OUTLINE

The system has been broken down into five sections comprising:-

* Entity Modelling
* Document Definition
* Transaction Details
* Normalisation
* Schema Generation

Within each section there is always a minimum amount of data that must be supplied. Beyond this, data entry is optional but not essential because the system has been supplied with enough information to create a basic schema. Obviously, the more data that is entered, the better the final schema generated will be.

3.4.1 Menu Screens

The programs are all selected from a series of menus, where all options relating to a certain function have been grouped together. A copy of the main menu screen is shown in Figure 3.3 The selection of one of these options will normally generate another

menu with the facilities available in that area, refer Figures 3.4 to 3.9 An option is selected by either entering the number or the word in capital letters.

3.4.1.1 Main Menu

S C H R O D E R D A T A B A S E D E S I G N S Y S T E M

1 MAINTENANCE Utilities
2 ENTITY Modelling
3 DOCUMENT Definition
4 ATTRIBUTE Maintenance
5 TRANSACTION Details
6 NORMALISATION
7 SCHEMA Generation

Figure 3.3

3.4.1.2 Maintenance Options

S C H R O D E R D A T A B A S E D E S I G N S Y S T E M

1 DATNAM Data Naming Conventions
2 PRTDNM Print Data Naming Conv

3 DICTWD Maintain Key Words

4 SYSCON System Contol Maintenance

Figure 3.4

3.4.1.3 Entity Modelling

S C H R O D E R D A T A B A S E D E S I G N S Y S T E M

1 ENTITY Model Definition
2 ENTITM Attribute Definition
3 ENTLST Entity Report
4 ENTATR Entity/Attribute Report
5 ENTREP Entity Model Report
6 ENTUSE Entity Usage Report

BATCH REPORT

7 ENTFUL Full Entity Report
8 ENTABR Abbreviated Entity Report
9 ENTREL Entity Relationship Report
10 ENTRAN Entity / Transaction Report
11 EAFULL Full Entity/Attribute Report
12 PRTENT Print Entity Report
13 PRTEAR Print Entity/Attribute Report
14 PRTEMD Print Entity Model Report

Figure 3.5

3.4.2 Document Definition

S C H R O D E R D A T A B A S E D E S I G N S Y S T E M

1 DOCDEF Document Definition
2 DOCLST Document Report
3 DOCERR Document Action Report

BATCH REPORT

4 DOCFUL Full Document Report
5 DOCABR Abbreviated Document Report
6 DOCLST Document List Report
7 DOCACT Document Action Report

8 PRTDOC Print Document Report
9 PRTACT Print Action Report

Figure 3.6

3.4.2.1 Attribute Maintenance

S C H R O D E R D A T A B A S E D E S I G N S Y S T E M

1 ATTREQ Attribute Enquiry
2 ATTRNT Attribute Maintenance
3 ATTRTY Attribute Type Definition
4 EXTPRC External Procedure Definition
5 GRPDET Group Name Definition

6 ATTRRP Full Attribute Report
7 ATTRCD Attribute Type Code Report

BATCH REPORT

8 ATRFUL Full Attribute Report

9 PRTATR Print Attribute Details
10 PRTACD Print Attr Type Code Report
11 PRTGRP Print Group Names Report

Figure 3.7

3.4.2.2 Transaction Details

S C H R O D E R D A T A B A S E D E S I G N S Y S T E M

1 TRLIST Brief Transaction Details
2 TRDET Full Transaction Details
3 TRENTU Entity/Transactio Report

4 TRVER Transaction Verification
5 TRBENT Entity/Transaction Report

Figure 3.8

3.4.2.3 Schema Generation

S C H R O D E R D A T A B A S E D E S I G N S Y S T E M

1 CRSCH Create Storage Schema
2 CRSTR Create Storage Schema
3 CRSUB Create a Subschema

4 SPSUB Specify a Subschema

5 DSPSC Display Schema

Figure 3.9

Now that the reader has seen an outline of the system, we will now proceed to look at these five sections in more detail.

CHAPTER 4

ENTITY MODELLING

4.1 BACKGROUND

In the previous chapter, the various methodologies for analyzing and designing a computer system were described. In recent years some organizations have placed considerable emphasis on selecting and using one of these methodologies. New computer systems are being designed daily, but only a few of them will actually use a database management system to hold their data. Instead most will rely upon conventional indexed or sequential files.

Attitudes are slowly changing towards database management systems. One of the objectives of a database is to model as closely as possible events that occur in the real world, and in order to achieve this, a model of the real world or system has to be constructed. The construction of models is not a new technique, except probably to the field of computing because for years engineers and mathematicians have been constructing models.

Consequently it was not very long before their application in the design of computer systems was recognised. Various types of model have been proposed, Chen suggested the Entity-Relationship model, Bachman the Entity-Role model, then there is the binary data model the Infological Data Model, or Codd's Relational Model.

The Entity-Relationship model was selected for this system, because it was the standard used in the organization sponsoring the research. It was not felt necessary to change the standard, because the Entity-Relationship model was ideally suited for this environment. Further justification for using this model came from a recent British Computer Society Conference on 'Data Analysis Methodologies' [BCS 82] where many of the speakers from industry spoke about how they used Entity Modelling in their environments.

The objective of Entity Modelling is to construct a model about the events that occur in the real world that are to be processed by the system. This is achieved by describing the real word in terms of :-

* Entities
* Relationships
* Attributes

An entity is a 'thing' that the business needs to hold information about e.g. people,places, things or events. Typical examples of these are Client, Branch, Policy and Payments respectively. A relationship exists between entities and is defined in terms of a name which expresses its meaning and a mapping which

indicates the number of entity occurrences participating in a relationship. The mappings available are:-

* 1:1 One Employee may only have one Company Car
* 1:M One Client may make many payments
* M:M A Client may have many policies and that policy may have more than one client

An attribute is defined as a characteristic of the entity that is used to describe it. For example, typical attributes of the Entity Client are Name, Age, Sex, Date of Birth etc.

To ascertain all of this information the Data Analyst must investigate and analyze the current systems and talk to its users. The final outputs from Entity Modelling will be:-

* the entity definitions
* the relationship definitions
* the entity diagrams

The Entity diagram, a typical example is shown in Figure 4.1 can then be shown to the users of the system for their approval. One of the reasons that Entity Modelling is so popular is that the Entity Diagram can be shown to non-computer people, who with only minimal explanation will quickly understand the diagram. It is very important at this early stage of the design to ensure that no aspect of the system has been overlooked or incorrectly modelled. Probably the best person to advise on this is the actual user of the system, but this will only be successful if the proposed computer design can

be explained simply.

One final reason for using the Entity-Relationship model was that the entity model diagram forms the basis of the translation to the physical CODASYL schema. This is achieved by directly translating entities into records and relationships into sets. The only instance where an intermediary translation is required is when converting a many to many relationship. e.g. In Figure 4.1, the relationship Client and Policy. It can be seen that the use of the Entity-Relationship model is ideally suited for a network database because it is merely a variation of the network model upon which the CODASYL network database is based.

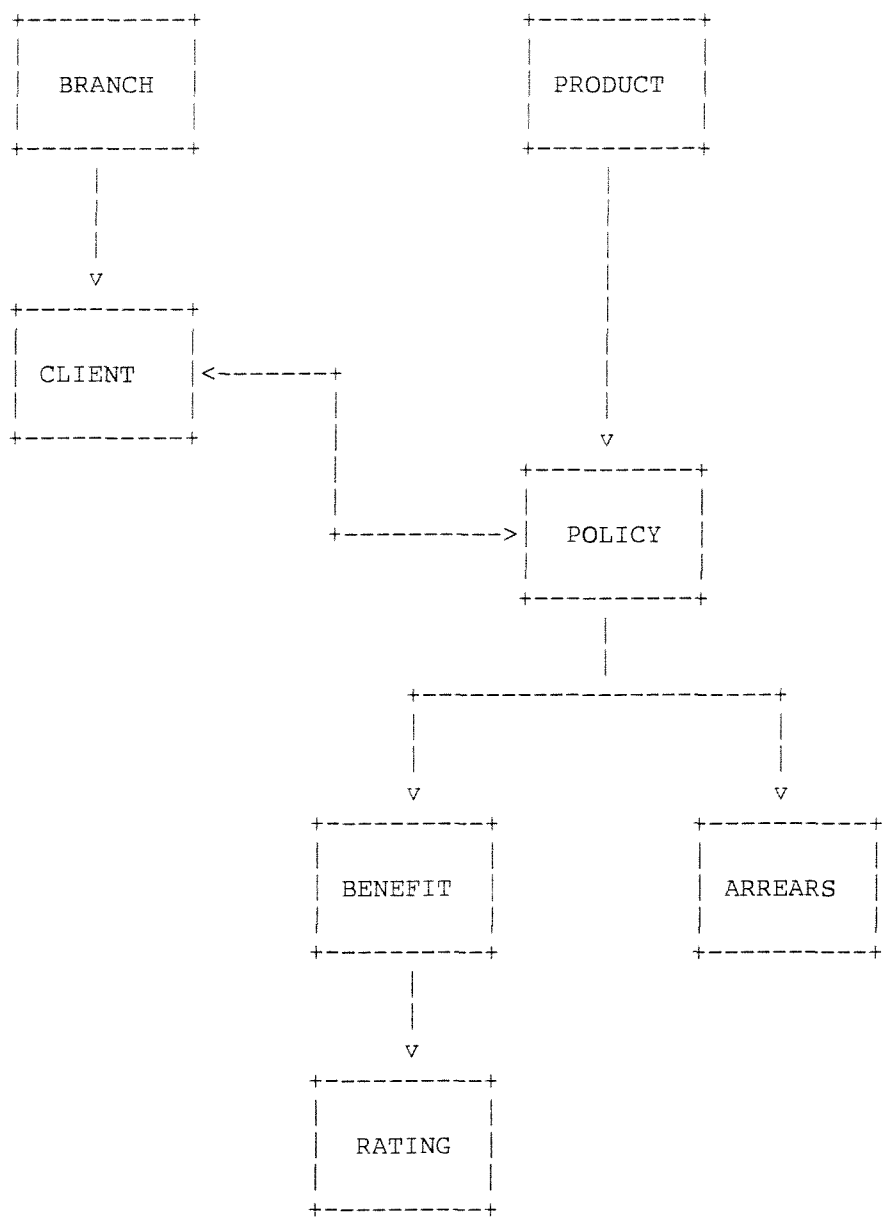


Figure 4.1

The analyst can begin to use this system once the construction of the Entity-Relationship model commences, which is essentially where Data Analysis begins. The following facilities are available to define the Entity Model:-

* Entity Model Definition
* Full Entity Definition
* Entity Reports
* Entity Model Diagram

Because this system has been defined so that only the minimum of data needs be input, the user must specify the entities using the Entity Model Definition. Remember that it is these entities that will form the basis for the records in the database. Once the entities have been designed the remaining facilities are then optional.

4.2 ENTITY MODEL DEFINITION

This program allows one to define the entity model without the need for a graphics terminal. The user is presented with a screen as shown in Figure 4.2 The screen is split into two sections. The left-hand side enables the user to specify two entities and the relationship that exists between them. On the right hand side of the screen, the relationships that the owning entity participates in are displayed. i.e. the entities that own it and the entities that it owns.

The following details may then be specified:-

* Owner Entity
* Member Entity
* Relationship between Entity
* A Description of the Relationship
* Starting Point Indicator
* Mandatory or Optional Relationship Indicator

The entity name is automatically defined in uppercase. The relationship between the entities must be defined in accordance with Chen's original Entity Relationship diagrams, that is 1:1, 1:M, M:1 or M:M. The system can also handle the definition of any entity that may own itself. This is achieved by defining a M:M relationship, where the owner and member entity are identical. An entity cannot be defined unless it participates in a relationship and any duplicate relationships are rejected. Next an optional description may be specified to describe the relationship. This is used on the entity relationship report to explain why the relation exists. e.g. An M:M relationship between the entities Client and Policy, where the description is 'may be linked to'. The details are reported as many Client may be linked to many Policy.

It was felt very important at this time to define if this Entity could be used as a Starting Point in the model. In the theoretical world starting points are not referred to in Entity Modelling, but nevertheless they do play a very important role. At the time the information is collected, it becomes apparent that

certain entities are always starting points for an event or process. If that information is retained now, then it can be used later to check that the entry points exist when one is attempting to validate a transaction against the model. This information then forms the basis for ultimately creating the physical access paths into the database. Of the systems reviewed, the only one that also allowed the definition of the starting point on the Entity Model was LBMS' Data-Mate.

The original definition of the Entity Relationship model didnot allow one to specify any constraints on a relationship. It was therefore suggested [SCHWEB] that the concept of a Total and Partial Relationship needed introducing. A total relationship is defined as one where the relationship between the two entities must exist e.g. a Policy must have a Client. Alternatively, a typical partial relationship is the one between the entities Benefit and Rating. A rating is not always applied to a benefit, therefore the relationship does not exist all of the time. Rather than use the terms total and partial, the CODASYL terms Mandatory and Optional have been used instead. This was done because it was felt that these terms were more self-explanatory . The relationship must be specified as M for Mandatory or O for Optional, the default is Mandatory.

Input of comprehensive information about the entity is deferred at this stage, because it was felt that this would unnecessarily delay the definition of the entity model, and the information was not needed at this time.

The following facilities are available:-

* Amend
* Delete
* Enquiry
* Insert
* Name Change

The Amend, Delete and Insert options are the typical maintenance options. The Enquiry facility may be used in one of two ways. Specify the owner and the member entity and the relationship details will be displayed along with the role of the owning entity on the right hand side of the screen. Alternatively, specify an entity name in the owning entity box and the role of this entity will be displayed on the right hand side of the screen. No matter what action is being taken the role of the owning entity is always displayed on the right hand side of the screen.

The Name Change facility provides the user with the capability of changing the name of an entity which will then automatically amend the name in each relationship in which it participates. e.g. To rename the Contract entity to Plan. Simply specify the old entity name in the Owner box and the new name in the Member box, the system then does the rest.

The relationship details are held in the data dictionary in the records ENTITY-REC which contains the entity details and the RELATION-REC which contains the relationship details (Refer Figure 3.2). An involuted relationship was defined between the records so

that the relationship of an entity owning itself could be defined.

It was decided to provide a utility to define only the entity model because this is one of the first tasks to be performed in Data Analysis and the output serves as the foundation for the physical design. It was also a feature that was distinctly lacking in most of the systems that were reviewed. Although this system lacks the graphics capability of LBMS Data-Mate, one can see that the definition of the model can be performed satisfactorily.

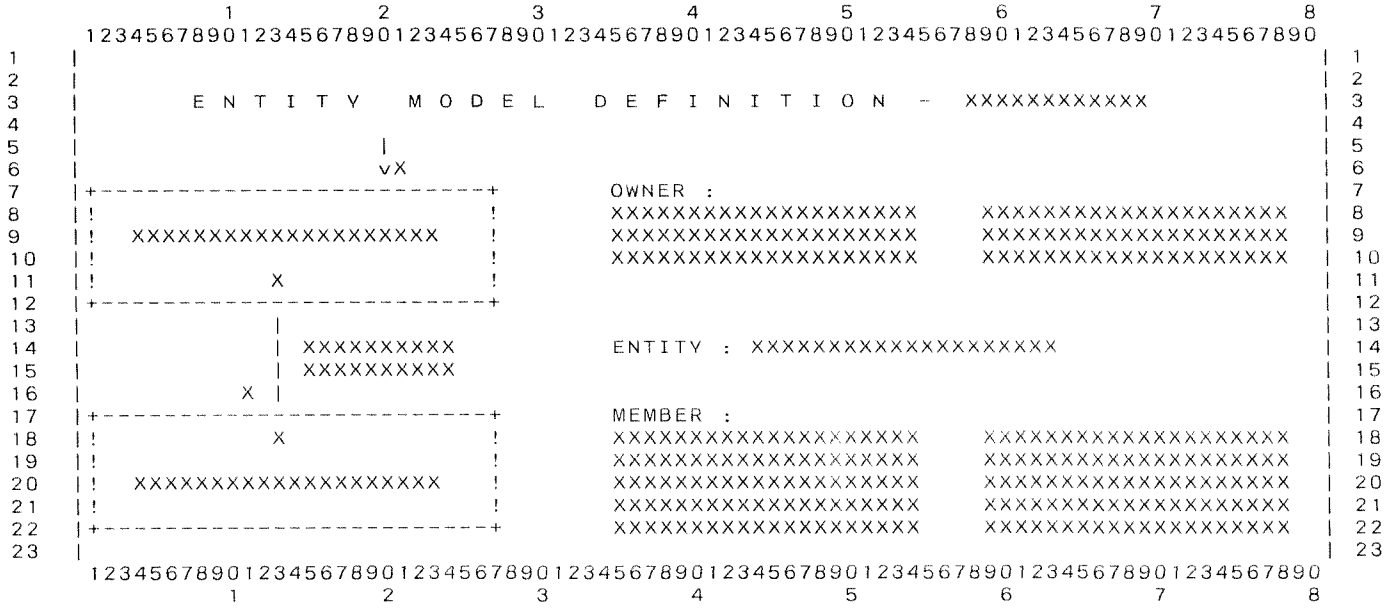

E N T I T Y M O D E L D E F I N I T I O N - XXXXXXXXXXXX
OWNER :
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX
X
XXXXXXXXXX
XXXXXXXXXX
ENTITY : XXXXXXXXXXXXXXXXXXX
MEMBER :
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX

Figure 4.2

4.3 FULL ENTITY DEFINITION

Once the Entity Model has been defined, this facility allows the entity details to be fully documented. Using this part of the system is optional because the basic entity details have already been defined.

Like the Entity Model Definition facility, the screen is once again split into two sections (Refer Figure 4.3) with the right hand side displaying the role of the entity and the left hand side being used for data entry. The following details may be defined:-

* Description
* Minimum Volume
* Maximum Volume
* Average Volume
* Percentage growth
* Area Name
* Group Identifier

A one hundred character description, describing the role of the entity can be specified. The volume data will be used to calculate the size of the database, and the percentage growth used to project the future database size. When the physical schema is constructed each entity will probably be translated into a record, therefore the Area Name is used to specify in which database area

the record will be placed. If this information is not specified then this will be updated by the system at Schema creation time.

To permit the specification of the high-level entity model and then low-level detail. The Group Identifier field indicates the high level entity for which the detailed information refers. For example on the high level diagram there may be an entity called DOCTOR, on the low-level diagram this may be expanded into the the entities DOCTOR, PATIENT and MEDICAL REPORT. The entities PATIENT and MEDICAL REPORT will have their group identifier field defined as DOCTOR, to indicate that they are the detailed definition of the entity DOCTOR. This lower level of detail on the Entity Model is often desirable in certain areas to avoid the overall Entity Model becoming unduly complicated. Currently both the high and low-level information is displayed on the entity diagram, in a future release the system will be modified to show only the level of detail that is required.

Using this facility only the details about an entity previously defined may be entered, this is to ensure that no inconsistencies occur in the data. Therefore, the only method available to create or delete an entity is to use the Entity Model Definition facility.

The following facilities are available :-

* Individual Entity
* All Entities
* Display
* Exit

The Individual Entity option allows amendment of an individual entities details. Upon completion of the changes, pressing GOLD C will accept the change and continue in the individual entity mode.

The All Entities option allows one to display/amend the details for all of the entities currently defined. The entities are automatically displayed in alphabetic order. As far as possible this system will try to assist the user and avoid the need for keeping lists and mountains of paper.

The Display option will display all of the detail about an entity. At all times the role of the entity is displayed on the right hand side of the screen.

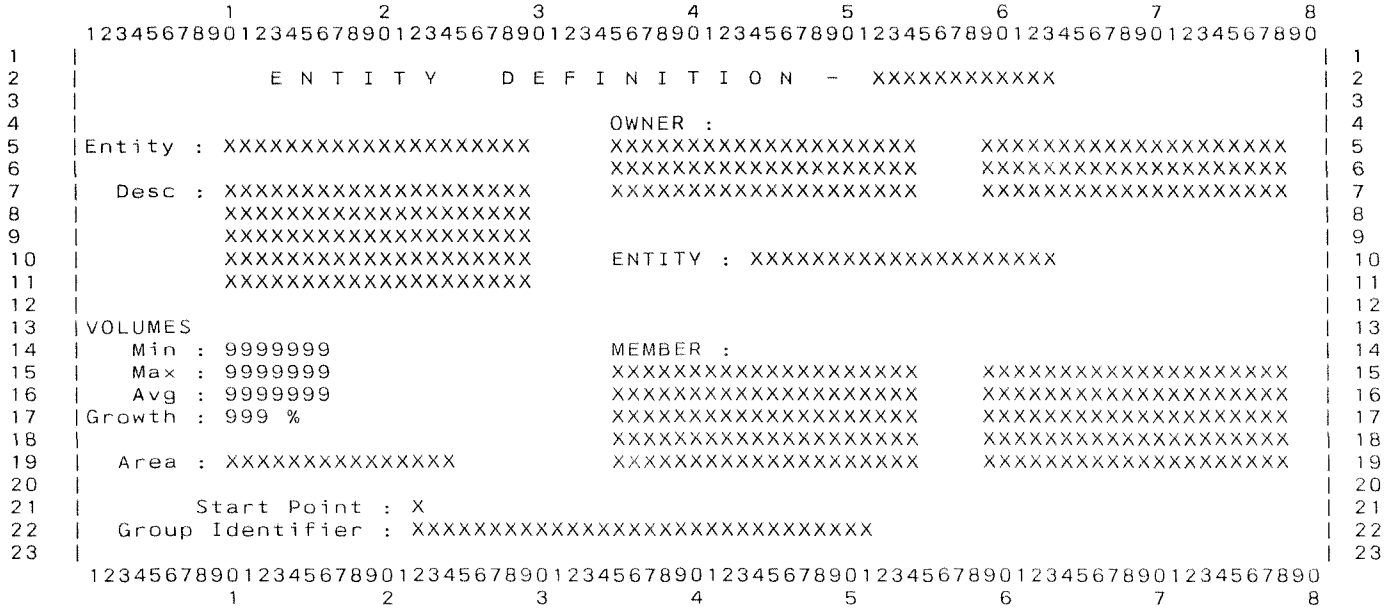

E N T I T Y D E F I N I T I O N - XXXXXXXXXXXX
OWNER :
Entity : XXXXXXXXXXXXXXXXXXXX
Desc : XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
ENTITY : XXXXXXXXXXXXXXXXXXXX
VOLUMES
Min : 9999999
Max : 9999999
Avg : 9999999
Growth : 999 %
MEMBER :
Area : XXXXXXXXXXXXXXX
Start Point : X
Group Identifier : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Figure 4.3

4.4 ENTITY REPORTS

The following entity reports are available within the system, these are selected either using the screen shown in Figure 4.4 or via a batch report. Examples of these reports are shown in Figures 4.5 , 4.6 , 4.7 and 4.8

* Full Report
* Abbreviated Report
* Relationship Report
* Transactions using an Entity Report

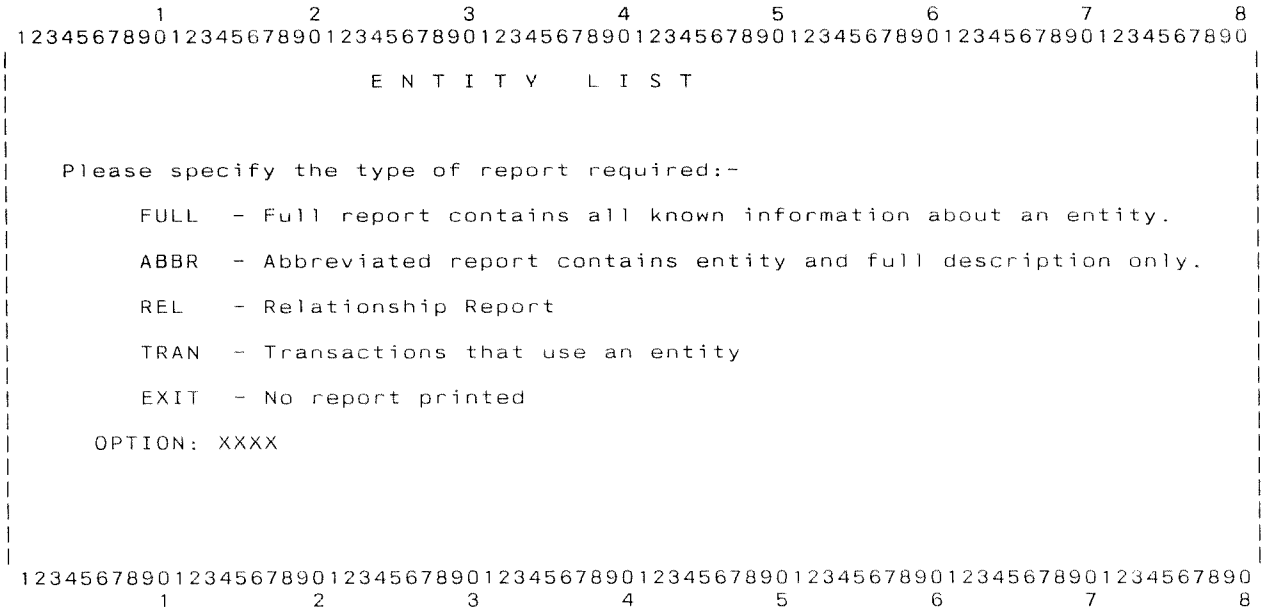

1 2 3 4 5 6 7 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
ENTITY LIST
Please specify the type of report required:-
FULL - Full report contains all known information about an entity.
ABBR - Abbreviated report contains entity and full description only.
REL - Relationship Report
TRAN - Transactions that use an entity
EXIT - No report printed
OPTION: XXXX
12345678901234567890123456789012345678901234567890123456789012345678901234567890
1 2 3 4 5 6 7 8

Figure 4.4

SCHRODER DATABASE DESIGN SYSTEM E N T I T Y L I S T 18-AUG-1986 PAGE: 1

	ENTITY	DESCRIPTION	VOLUMES MIN	MAX	AVG	GROWTH	AREA
	ARREAR	A payment due on a policy that has not yet been r eceived.		25000	5000	2%	POLOTHDATA
	BENEFIT	Any statement in the policy schedule which gives t he clients right to a benefit	1	360000	250000	20%	POLMAINDATA
S	CLIENT	A person(s)/body which has a legal interest in a p olicy or proposed policy.	1	300000	175000	%	
S	CONTRACT	A reference under which a group of policies are ad ministered together for one client.	1	60000	25000	18%	
S	DOCTOR	Medical Examiner who examines a client	1	10		1%	POLOTHDATA
			Group Identifier: DOCTOR				
	FUND	The investment pool into which a proportion of cli ent's contributions are received.	1	500000	300000	20%	
	MEDICAL REPORT	A report issued by a Medical Examiner declaring the state of health of the Client		100		%	POLOTHDATA
			Group Identifier: DOCTOR				
S	POLICY	A legal document drawn up that defines acontractua l obligation between the company and the client.	1	300000	250000	20%	
	RATING	An adjustment to company's standard benefit/premiu m calculation. Used when risks are greater.		25000	15000	5%	
	UNIT	The equal share of aunit linked fund.	1	1000000		20%	

Figure 4.5

SCHRODER DATABASE DESIGN SYSTEM E N T I T Y L I S T 9-MAR-1986 PAGE: 1

ENTITY	DESCRIPTION
ADDRESS	
AGENT	
AUTHORITY	The authority to make a payment of periodic premiums.
BENEFIT	Any statement in thepolicy schedule which gives the clients right to a benefit.
BRANCH	An area office from which direct sales agents and consultants operate.
BROKER	An independent professional advisor or insurance broking company.
BROKER CONSULTANT	An employee of the company who introduces the company and its products to potential brokers.
CLIENT	A person(s)/body which has a legal interest in a policy or proposed policy.
COMMISSION PAYMENT	
FINANCIAL CONSULTANT	
FUND	The investment pool into which a proportion of client's contributions are received.
FUND DISTRIBUTION	
INVESTMENT	The asset into which a proportion of the fund is invested by the fund manager.
PENSION SCHEME	A pension scheme approved by the Superannuation Funds Office for membership by employees/directors.
PLAN	A unique definition for a type of policy issued by the company.
POLICY	A legal document drawn up that defines a contractual obligation between the company and the client.
PRODUCT	A series of plans which are issued by the company, grouped together for financial & marketing.
RATING	An adjustment to company's standard benefit/premium calculation. Used when risks are greater.
RETAINED BENEFIT	

Figure 4.6

One	AGENT	is renumerated on	Many	COMMISSION DATE
One	BANK ACCOUNT	may have facility of	Many	LOAN
One	BANK ACCOUNT	may have in force	Many	AUTHORITY
Many	BANK ACCOUNT	do business with	Many	CLIENT
One	BANK BRANCH	is responsible for	Many	BANK ACCOUNT
One	BENEFIT	may be adjusted by	Many	RATING
One	BRANCH	achieves sales	Many	FINANCIAL CONSULTANT
One	BRANCH	achieves sales via	Many	AGENT
One	BRANCH	employs	Many	BROKER CONSULTANT
Many	CLIENT	may be linked to	Many	UNIT TRUST CONTRACT
Many	CLIENT	do business with	Many	BANK ACCOUNT
Many	CLIENT	may be linked to	Many	POLICY
One	CLIENT	may choose to have	Many	ADDRESS
One	CLIENT	may have	Many	RETAINED BENEFIT
One	COMMISSION DATE	triggers payment of	Many	COMMISSION PAYMENT
One	CURRENCY	is expressed in	Many	EXCHANGE RATE
One	FUND	may have units in	Many	CURRENCY
One	FUND SERIES	is divided into	Many	FUND UNIT TYPE
Many	LOAN	may legally link to	Many	POLICY
One	PLAN	groups together	Many	POLICY
One	POLICY	may have attached	Many	UNIT HOLDING

Figure 4.7

ENTITY	TRANSACTION
ARREAR	Direct Debit
BENEFIT	Direct Debit
BRANCH	Application
BROKER	Application
CLIENT	Application
FUND	Direct Debit
PERIODIC PREMIUM	Direct Debit
POLICY	Application Direct Debit
UNIT	Direct Debit

Figure 4.8

4.5 ENTITY MODEL DIAGRAM

The final facility available is the generation of an Entity Model Diagram. As was stated earlier this system had to be developed using existing hardware. Because no plotters or graphics terminals were available from which a plot could be produced, a simple Entity Model Diagram as shown in Figure 4.9 is generated. The diagram generated is primitive compared with systems such as LBMS Data-Mate. However, it does provide an outline of the system, and at least this system provides more information than most of the Data Dictionaries reviewed. Only a few of the systems reviewed allowed the specification of the Entity Model and not all of those attempted to display the model.

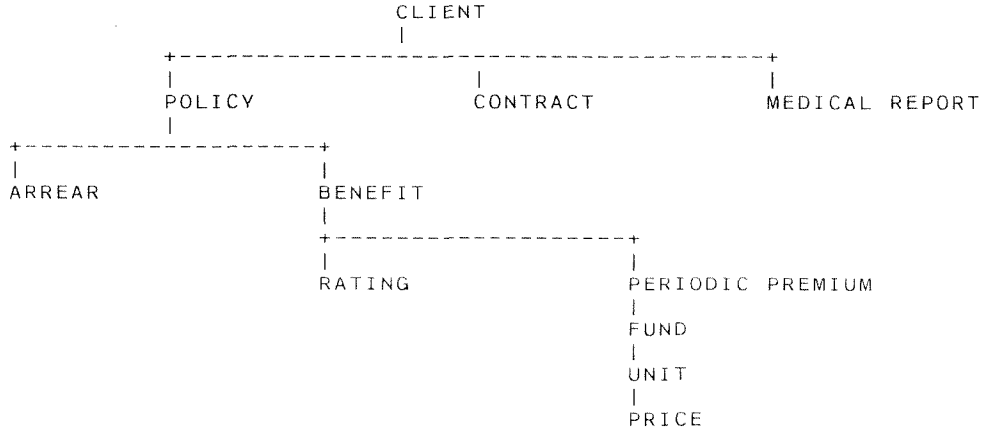

CLIENT
POLICY
CONTRACT
MEDICAL REPORT
ARREAR
BENEFIT
RATING
PERIODIC PREMIUM
FUND
UNIT
PRICE

Figure 4.9

CHAPTER 5

DOCUMENT AND ATTRIBUTE DEFINITION

5.1 OVERVIEW

Once the Entity-Relationship model has been defined, Entity Modelling is not complete until all the attributes of the Entities have been defined. To determine this information, the analyst has to investigate the current procedures and discuss the requirements of the new system with the users. As a result of this investigation, the inputs and outputs of the system will be identified, for example these could be a manual form, a report or a screen on a vdu. Once the Data Analyst has identified all the inputs and outputs of the current or proposed system, these documents are scanned, and the attributes on them are identified, and allocated to an entity. The importance of using this technique is that it is then possible to identify the source of every attribute. Otherwise it is possible that later in the design, an attribute may be discovered whose source is unknown. Rather than omit this attribute from the database, or create considerable work going over old ground trying to discover its source, the attribute

will often be included in the design to save time. Although the inclusion of this attribute does not invalidate the model, it can lead to ambiguities or redundant data being included in the final design. Therefore one of the objectives of this system is to avoid this type of confusion, and prevent wasting valuable disk space. It is anticipated that by forcing the analyst to specify the source of all data, duplicate fields will be quickly identified, and the reason for the inclusion of every attribute will be readily available to resolve any queries that may arise. Consequently, only the attributes that are really needed will be included in the design, which all helps to create a computer system that does not contain any redundant data.

The approach taken by this system to define the attributes of the entities is very different to the conventional Data Dictionary and Design tools that were discussed previously, in so much as the user may specify any type of document that may be used in the system. A document is defined as as any input or output from or to the system, such as screen layouts, report formats, computer records or even manual forms. From these different types of documents, only limited information about each attribute need be entered into the system. Then if required, more detail may be supplied later. Using this facility it is not possible to create the actual layout of the document from this information, only the attributes that appear on the document may be identified. By splitting the task of attribute definition into two separate functions, the analyst is forced to specify the basic details about

each attribute before design continues, without having to waste time entering the detail, which can be supplied later. This should not be seen as duplicating effort, because it ensures that a minimum quantity of data is input, without unnecessarily delaying the designer with the tedious detail at this stage.

It is only at this stage of development that the Data Dictionary systems that were reviewed, began to offer some of the facilities that are available within this system. Using a typical Data Dictionary system it is possible to define record layouts and all of their attributes using a rigid syntax as per figure 2.3. Normally screen formats, file and database definitions may be held in the dictionary, but not details about manual forms. The advantage of the Data Dictionary systems is that they have been designed so that the definition may be included directly into a program. Therefore when a screen layout is held in the data dictionary, the actual format of the screen is retained so that it can be built quickly. Since this system has been built more as a design tool than a dictionary system interfacing to programs, less emphasis has been placed on retaining the actual format of the documents.

Within this area the following facilities are available:-

* Document Definition
* Document Report
* Document Action Report
* Attribute Maintenance
* External Procedure Definition

* Attribute General Enquiry
* Attribute Report
* Attribute Type Code Definition
* Computer Name Codes Definition

As stated previously, if a 'quick' database design is required, then the only compulsory stages that must be completed within this area are that of Document Definition and the generation of an error free Document Action Report. It is the Action Report which identifies any inconsistencies in the model that might jeopardize the creation of the physical schema. Upon completion of this part of the system, the task of Entity Modelling will be complete because all of the attributes will have been defined.

Let us now look at the various facilities available within this part of the system, to see how they compare with the products that are currently available.

5.2 DOCUMENT DEFINITION

The Document Definition Facility enables each document and its basic attributes details to be defined. Rather than burden the Data Analyst with long data entry sessions, which will do nothing for the system except discourage people to use it, it was decided to provide a facility which allowed only the basic details about an attribute to be defined, comprehensive information may be supplied later. Whilst it could be argued that all of the data should be

supplied at this stage, it was felt that some of this information may not yet have been determined. This is a different approach to the standard Data Dictionary system. For example using the ICL Data Dictionary System, when a record is defined there is no minimum quantity of information that must be specified. Therefore only the attribute name may be entered, or very detailed information.

Instead this system presents the user with a screen as shown in Figure 5.1., which means that it can be used by either the Data Analyst or a non-computer person filling in the screen from some hand-written sheets. The screen is split into two, the left hand side is used for inputting details about the current attribute and the right hand side contains a list of the last twelve items that have been defined on this document. Each document is then given a name and a type, which is then used for all subsequent references. If the type is specified as 'M' then a manual form has been defined. The system does not really care at this stage what type of document is being defined, what it needs to know is everything about an attribute, and this information is what the analyst is being forced to provide.

To clarify the use of some of the fields, the Attribute Text Name which is compulsory, is the name by which the attribute is usually known and the computer name is the name usually given to the attribute on a computer based file. Surprisingly some of the Data Dictionaries did not allow the specification of the computer name, the text name was the only one that was permitted. In this system it is optional, but the need for the computer name is recognised

because it is often the only name by which application programmers can refer to an attribute and analysts often do not know the computer name, only the text name.

The Picture format is validated so that it conforms to the conventions used in COBOL, that is numeric fields are defined using 9's and COMP, and text fields with an X, this is the same as most of the Data Dictionary systems. During the course of the investigation, certain attributes will keep appearing as keys. It therefore seemed sensible to specify if an attribute formed part of the key, whether it is a Primary or a Candidate key, and if it forms part of a composite key, then its position within the key. Although the ability to specify whether the attribute is a Primary or Candidate key is not unusual, the specification of the position of the attribute within the key is not common. Please refer to Table 5-A for a complete list of all the information that may be defined.

One of the main purposes of this part of the system is to allocate the attribute to an entity. The only Data Dictionary system that allowed the entity model to be defined was ICL DDS. Products such as DATA-MATE will define the entity model, but the attributes are identified manually and are then entered into the system. Using this system, if the entity is not specifically entered, then the system will automatically try to determine which entity it should be, from the attribute name. The decision is made by searching the dictionary for any entity whose name matches the first part of the attribute name. In practise this approach has been found to work very well and there has been a very high success

rate using this simple technique. Any attribute that cannot be allocated to an entity will be reported later.

The usual maintenance options are available.

* Insert
* Amend
* Delete
* Display

The Insert Option permits the insertion of new documents and their attributes or the addition of new attributes to a document. Whenever an attribute is being added to a document, if it has previously been defined then there is no need to enter all of the details again. The system will detect the existence of the attribute and ask if this item is to be added to the document. Simply confirm that this is the case, and the current definition of this attribute will be included on this document. This not only saves data-entry time but maintains the consistency within the model. If the details about an attribute are inadvertingly entered again and they differ from the original entry, then this will be reported as an error. It should be noted that to maintain the consistency in the model, this is the only place where a new attribute may be defined.

The Delete option, permits either deleting the entire document or individual attributes on the document. The Amend and Display options will amend/display all of the attribute on a document.

The storage of this information in the Data Dictionary, refer figure 3.1 and 3.2 makes use of the network structures to implement the relationship. The attribute details are held once only in ATTRIBUTE-REC. Then for each document specified,a FORM-REC is created, and for every attribute that appears on that document a FORM-TYPE record is created that serves as the junction record between the document and the attribute. Using the junction record to implement the many-to-many relationship exists between document and attribute, means that it is possible to only specify the attribute detail one, and then list all of the documents on which this attribute appears. This is one of the advantages that results from using a network database to implement the data structures.

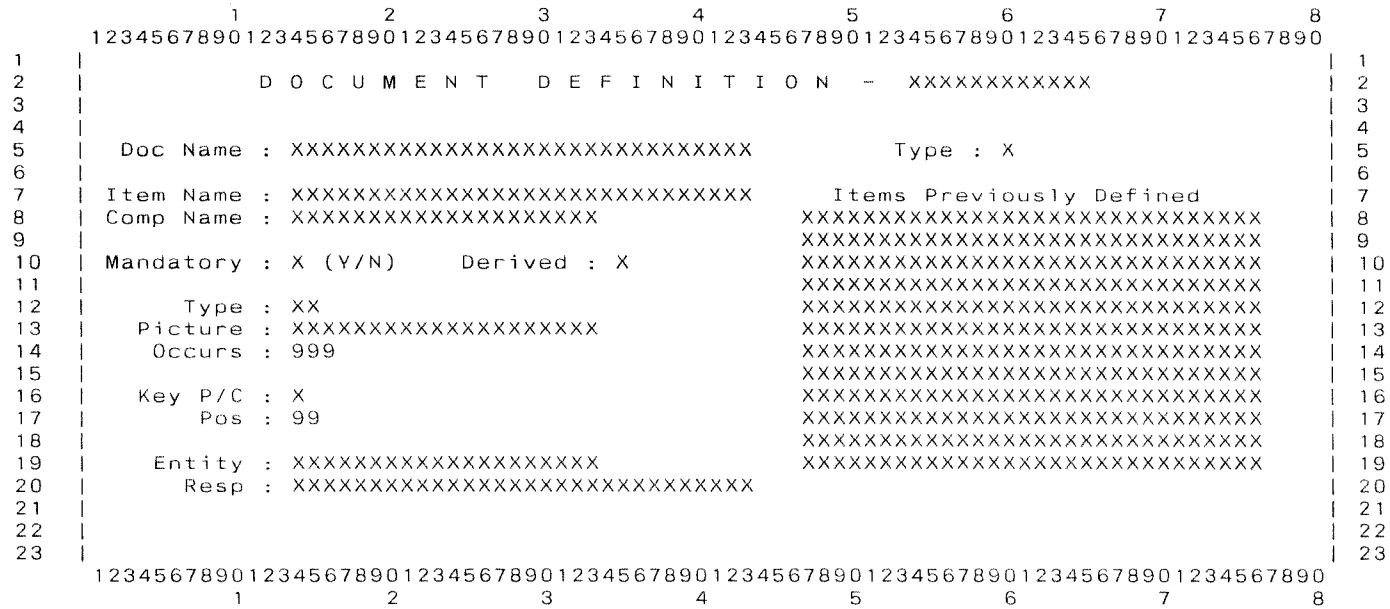

1 2 3 4 5 6 7 8
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
D O C U M E N T D E F I N I T I O N - XXXXXXXXXXXX
Doc Name : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Type : X
Item Name : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Items Previously Defined
Comp Name : XXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Mandatory : X (Y/N)
Derived : X
Type : XX
Picture : XXXXXXXXXXXXXXXXXX
Occurs : 999
Key P/C : X
Pos : 99
Entity : XXXXXXXXXXXXXXXXXX
Resp : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Figure 5.1

5.3 DOCUMENT REPORTS

No system is complete without the ability for the user to produce a number of reports. Compared to systems like Data Catalogue 2 and IDD which has over 70 reports, the number of reports available in this system is small. However, the emphasis has been placed on producing error reports or informational ones that can be used to identify problems with the system. Although this system is essentially a design tool, unlike many of its so called competitors it is user-friendly and with its formatted screens, there is less of a requirement for mountains of paper to be printed because the information is available on the screen. Because of the strict validation, often the system will not accept data until it is correct, nevertheless, several reports are available, and samples of these are shown on the following pages.

* Document Reports

  - o Full Document Report (Figure 5.2)

  - o Abbreviated Document List comprising of the Document Name and its Attributes (As per Figure 5.2 without the detail)

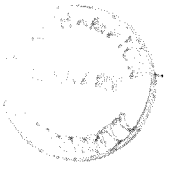
  - o List of Document Names and their types only (As per Figure 5.2 without the detail)

* Entity/Attribute Report

  - o All Entities (Figure 5.3)

  - o Individual Entity

* Document Action Report (Figure 5.4)



DOCUMENTS	TYPE	ATTRIBUTE		COMPUTER NAME		MAND	DERV	TYPE	PICTURE
Application	M								
		Client Name Key: P Pos: 1	CLIENT			Y	N	TX	X(30)
		Age	CLIENT	AG	Form	N	N	NR	999
		Sex	CLIENT	SEX		Y	N	CD	X
		Address	CLIENT	ADDR		Y	N	TX	X(30)
		Date of Birth	CLIENT	DOB		N	N	NR	9(8)
		Product Code		PROD_CD		Y	N	CD	X(8)
		Payment Amount	PERIODIC PREMIUM	AMT_PAY		Y	N	AM	S9(7)V99
		Payment Frequency		PAY_FREQ		Y	N	CD	99

Figure 5.2

ENTITY	ATTRIBUTE	PICTURE	OCCURS	KEYS
ARREAR				
BENEFIT				
CLIENT				
	Client Name	X(30)		P 1
	Age	999		
	Sex	X		
	Date of Birth	9(8)		
	Client Reference	X(8)		P 1
	Address	X(30)	5	
CONTRACT				
DOCTOR				
FUND				
	Fund Code	999		P 1
MEDICAL REPORT				
PERIODIC PREMIUM				
	Payment Amount	S9(7)V99		
POLICY				
	Policy Number	X(9)		P 1
	Policy Start Date	9(8)		
	Policy End Date	9(8)		

Figure 5.3

5.3.1 Document Action Report

When the Data Dictionary systems were being reviewed, what seemed to be lacking from most of them was some form of error report that would identify missing items. Because they were designed as a repository, rather than as a design tool, this area had been overlooked, although products like Data Catalogue 2 could be interrogated using its own query language. Therefore imagine the problems with a system with over 900 attributes, it can take hours to identify all of the missing fields, and then there is no guarantee that one has not been overlooked. Therefore it was decided to introduce the Document Action Report, an example is shown in figure 5.4, which serves as an important checkpoint in the system. Its function is to reassure the Data Analyst that the next stage of design can be attempted without essential data being missing, because each attribute is scanned and the following errors are reported.

* All Attributes That Are Not Allocated to an Entity
* Invalid Primary or Candidate Key Positions
* Attributes Referenced in a Procedure that do not exist
* No Attribute Type Code Specified
* No Picture Format
* No Description for an Attribute

Since the report plays a key role in the development of our schema an indicator is maintained in the system control record to record whether the Action Report has been generated, and if so whether it completed with any errors. At schema creation time, an error will be reported if an error free document action report has not been generated. It was surprising that none of the systems reviewed maintained any record of whether the error reports had identified problems earlier in the design. This information is vital to the Analyst and Designer, to clarify that all of the necessary steps have been completed successfully.

SCHRODER DATABASE DESIGN SYSTEM D O C U M E N T A C T I O N R E P O R T 26-AUG-1986 PAGE: 1

DOCUMENT	TYPE	ATTRIBUTE	
Application	M		
		Client Name	ATTRIBUTE NOT ALLOCATED TO A ENTITY
		Age	ATTRIBUTE TYPE CODE NOT SPECIFIED
		Age	ATTRIBUTE NOT ALLOCATED TO A ENTITY
		Sex	ATTRIBUTE NOT ALLOCATED TO A ENTITY
		Address	ATTRIBUTE NOT ALLOCATED TO A ENTITY
		Date of Birth	NO DESCRIPTION SPECIFIED
		Date of Birth	ATTRIBUTE NOT ALLOCATED TO A ENTITY
		Product Code	NO PICTURE CLAUSE SPECIFIED
		Product Code	NO DESCRIPTION SPECIFIED
		Product Code	ATTRIBUTE NOT ALLOCATED TO A ENTITY
		Payment Amount	NO DESCRIPTION SPECIFIED
		Payment Amount	ATTRIBUTE NOT ALLOCATED TO A ENTITY
		Payment Frequency	NO DESCRIPTION SPECIFIED
		Payment Frequency	ATTRIBUTE NOT ALLOCATED TO A ENTITY

Figure 5.4

5.4 ATTRIBUTE MAINTENANCE

Once the attributes have been defined using the Document Definition Facility, the next stage of design can now be attempted. Alternatively, or even after the design has been agreed, detailed information about these attributes may be entered. The decision to separate the input of the detailed information and make this step optional, was done because enough information has already been defined in the previous steps to create a Schema. This system is essentially a design tool, not a Data Dictionary, although the detailed information retained about the attributes is as comprehensive as many of the Data Dictionary systems, where this system is lacking is in what can be done with the information. For example, because this system is essentially a design tool, it is not possible for an application program to reference the system directly to extract record or screen layouts as per systems such as ICL DDS.

Despite this disadvantage, one of the real advantages of this system is the use of the formatted screens for displaying and entering the information. A major objective was to make the system user-friendly and easy to use, because a number of the systems reviewed were lacking in this area. Two screens are used, the first as per figure 5.5, comprises all of the basic attribute information and the continuation screen, refer figure 5.6, is used to display

additional information comprising of the documents that this attribute appears on, its computer names and synoynms. The later screen contains a scrolled area, so that the user is not restricted to thirteen lines of information. Please refer to Table 5-B for a complete list of the information retained about an attribute.

Using this facility, all of the information displayed may be changed except the document list. This restriction is imposed to prevent the inclusion of attributes, without previously defining the document on which it appears. Due to the structure of the Data Dictionary, when the details about an attribute are amended, then because the attribute is only defined once, the amendment will apply to every document on which this attribute appears. The separate maintenance facility only exists so that the updating of the model is restricted to the Data Analyst and Designer, all other users of the system must use the General Enquiry Facility to retrieve the same information, although the presentation screens are identical. The following facilities are available:-

* Amend
* Display
* External Procedure
* Name Change

The Name Change Facility was introduced to enable the name of an attribute to be changed without the need to specify all of the information again across all of the documents used. This is very useful when a typing mistake occurs.

The External Procedure option allows the user to see the definition of external procedures.Please refer to section 5.4.1 for further information regarding this facility.

```
             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    |                    I T E M   D E F I N I T I O N                 Page - 1     |
    |                                                                               |
    |   Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX       Comp Nm: XXXXXXXXXXXXXXXXXXXX    |
    |                                                                               |
    |   Desc: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                    |
    |         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                    |
    |         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                    |
    |         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                    |
    |                                                                               |
    |Picture: XXXXXXXXXXXXXXXXXXXX        Just: X   Occurs: 999   Type: XX          |
    | Status: X                        Derived: X       Key: X      Pos: 99         |
    |                                                                               |
    |Default: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX          |
    |Rge Chk: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX          |
    |                                                                               |
    | Entity: XXXXXXXXXXXXXXXXXXXX      Security Level Read: XX   Write: XX         |
    |  Table: XXXXXXXXXX                                                            |
    |   Resp: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                                        |
    | Create: XXXXXXXXXXXXXXX      Data Protection Act Classification : XXXX        |
    |                                                                               |
    |    Ins: XXXXXXXX     Last Amended: XXXXXXXX                                   |
    |                                                                               |
    |                                                                               |
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
```

Figure 5.5

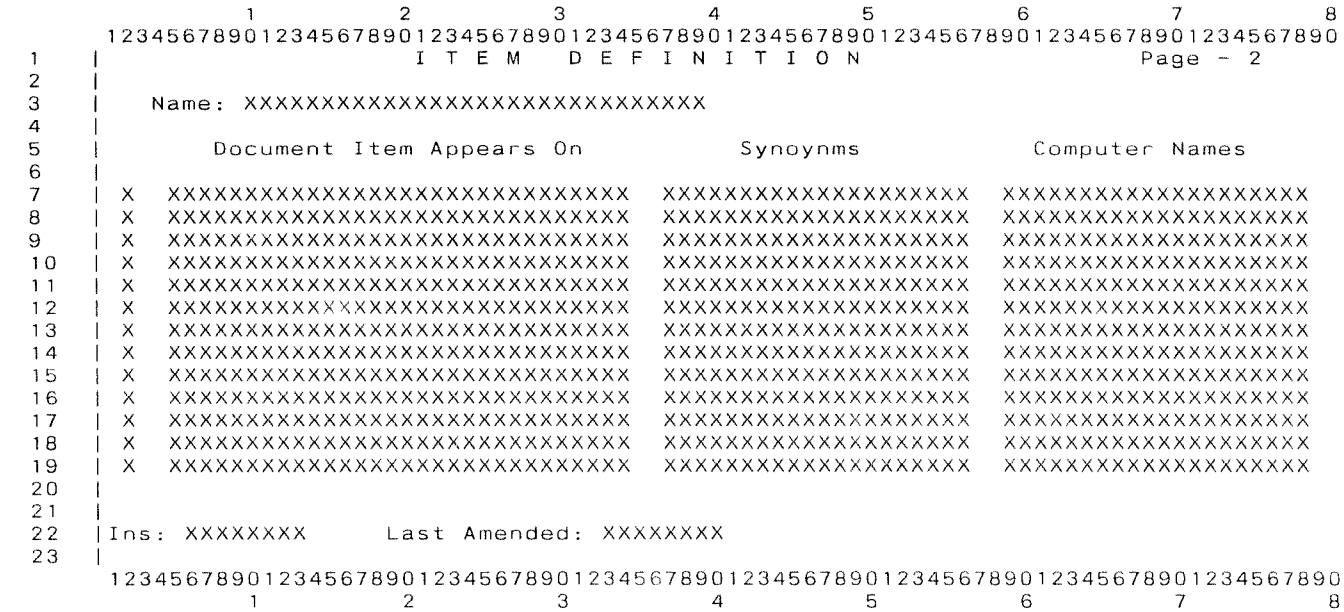

ITEM DEFINITION
Page - 2
Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Document Item Appears On
Synoynms
Computer Names
Ins: XXXXXXXX
Last Amended: XXXXXXXX

Figure 5.6

5.4.1 External Procedure Definition

Quite often an attribute can take a range of values, therefore it is very important to retain this information not only for reference, but also for checking the data values. This information is retained in the system in the Attribute Range Check field, provided it can be specified in less than sixty characters, otherwise, an external procedure must be defined, and the name of the procedure will be retained here instead. A procedure name is easily identified because the first character in column one is a colon. A number of the Data Dictionaries provided this facility, but they also went one step further because they allowed the applications to check the dictionary at run-time to see if the value of the attribute was within the required range.

Within this facility all of the values may be specified in a scrolled area as per figure 5.7, so the user is not restricted to the amount of space available on the screen, and since there is no validation of this field, the format of the data is entirely dependant on the user. If required, the values may be included in the schema, but to do this the user is restricted to defining the procedure using the syntax required by the schema definition

language. By providing a free-format area, the user is not restricted to the style of data-entry, which makes the system more flexible to use, and this is a particular advantage if the details are required for information purposes only.

Each procedure must have a procedure name, and the attribute to which this procedure refers using either the Text or Computer Name. Once the specification of the procedure is complete, the attribute details are automatically updated with the procedure name. It is not possible to specify the procedure name in Attribute Maintenance, because if the procedure was omitted this would invalidate the model. The reasoning behind including this facility within the maintenance option is that it enables the user to define an attribute and then immediately afterwards define the External Procedure. The details of an external procedure can always be seen without having to leave the facility and the information is always reported when the complete details of an attribute are printed.

At the outset of design the user must specify whether the external procedure details are to be included in the schema, this information is held in the CONTROL-REC. If they are to be included in the schema, then the detail must be defined in a format that can be included in the Schema directly, otherwise the format is at the discretion of the user. The details of the external procedure are held in a separate record called EXT-PROC-REC and only one external procedure may exist for an attribute, refer to Figure 3.2.

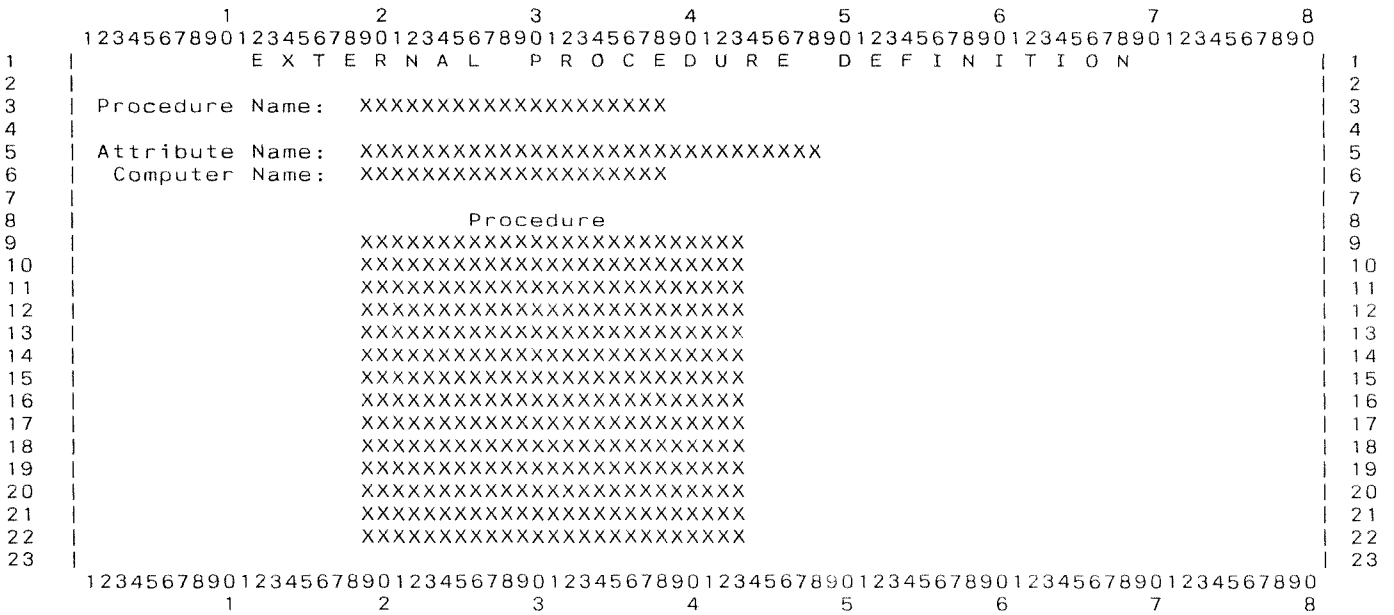

1 2 3 4 5 6 7 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
EXTERNAL PROCEDURE DEFINITION
Procedure Name: XXXXXXXXXXXXXXXXXXX
Attribute Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Computer Name: XXXXXXXXXXXXXXXXXXX
Procedure
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX
12345678901234567890123456789012345678901234567890123456789012345678901234567890
1 2 3 4 5 6 7 8

Figure 5.7

SCHRODER DATABASE DESIGN SYSTEM I T E M D E F I N I T I O N

Name: Policy Status

Desc: A code to indicate the current status of the
policy. The code falls into one of 3 groups:-
Proposals In Force Risks Terminations

Picture: 99 Occurs: 0 Just: Status: P
Type: CD Derived: Key: Pos: 0

Default:
Rge Chk: 00 = Record Present 01 = Normal Proposal
02 = Proposal Cancelled 03 = Postponed Proposal
04 = Declined Proposal 05 = NPW
10 = In Force 20 = Recission
21 = CFI 22 = Settled
23 = Settled 30 = Surr Notified
31 = Surr Settled 32 = Auto Surrender
40 = Death Notified 41 = Death Settled
50 = Fully Paid Up 51 = Reduced Paid Up
52 = Auto Paid Up 60 = Maturity Notified
61 = Maturity Settled 70 = Expired
80 = Replaced 90 = Lapsed No Value

Resp: Table:
Create: Code Controller Security Level Read: Write:

Entity: POLICY Data Protection Act Class:

Document Item Appears On	Synoynms	Computer Names
A201_REC	A501 A501_POL_STAT	A201_POL_STAT
A501_REC		A501_POL_STAT

Ins: 05/02/86 Last Amended: 07/02/86

Figure 5.8

SCHRODER DATABASE DESIGN SYSTEM I T E M T Y P E R E P O R T 1-SEP-1986 PAGE: 1

ITEM TYPE	ATTRIBUTE	PICTURE	OCCURS	KEYS
TX Text				
	Address	X(36)	28	
	Bank Account Name	X(18)		
	Forenames	X(40)		
	Fund Abbreviated Description	X(10)		
	Fund Description	X(30)		
	Initial	X		
	Mailing Name (Client's Title)	X(36)	7	
	Message	X(40)		
	Short Name	X(4)		
	Style of address e.g. Sir , Ma	X(36)	7	
	Surname	X(72)		
	Surname of First Life Assured	X(40)		

Figure 5.9

5.5 ATTRIBUTE GENERAL ENQUIRY

If this system is to be used by people other than the design team, it must have a comprehensive enquiry facility to interrogate the data. What was surprising was that a number of the Data Dictionary systems had enquiry facilities, but most required using a rigid syntax for interrogating the data. The problem with this is that casual users of the systems such as application programmers, will probably have difficulty extracting the information, because they are not familiar with the syntax.

To overcome this problem, the rigid syntax has been abandoned and a screen in presented as per Figure 5.10 where the user may search on:-

* Full Attribute Text Name
* Full Attribute Computer Name
* Partial Attribute Text Name
* Partial Attribute Computer Name

The disadvantage with this technique is that it is not possible to perform selective searches on the data, which some of the other systems could offer. To overcome this problem, since all of the data is held in a database, the DBMS query language can be used to select the required information. Unfortunately, this does

mean that the information cannot be presented in the screen format, but at least the details can be supplied.

It may be surprising to learn that some of the systems reviewed did not even provide facilities to interrogate the data using only part of the key. It was felt that this was an important facility to provide, because very often the complete name is not known, or there is a requirement to identify all of the attributes that make reference to say the word 'bank'. Remember, that although this system is essentially used by the Data Analyst and Designer, an important benefit is the availability of this information to all personnel involved in developing computer systems for this database.

When a partial key search is requested, a list is compiled of all the attributes that contain the part of the key specified. Using a scrolled area the list of attributes is displayed and each attribute is given a number. To view the details of an attribute, simply enter the number and the details will be displayed on the screen shown in figure 5.5, or the continuation screen (figure 5.6). Once an attribute has been identified the options available are:-

* Display
* External Procedure (Refer section 5.4.1)
* Print (Figure 5.8)

In the installation where this system was being developed. The application programmers welcomed the ability to search on the computer name and they especially liked the partial key searching facility. The screen presentation makes the system easy to use,

therefore enquiries can be performed easily and consequently people started using the system.

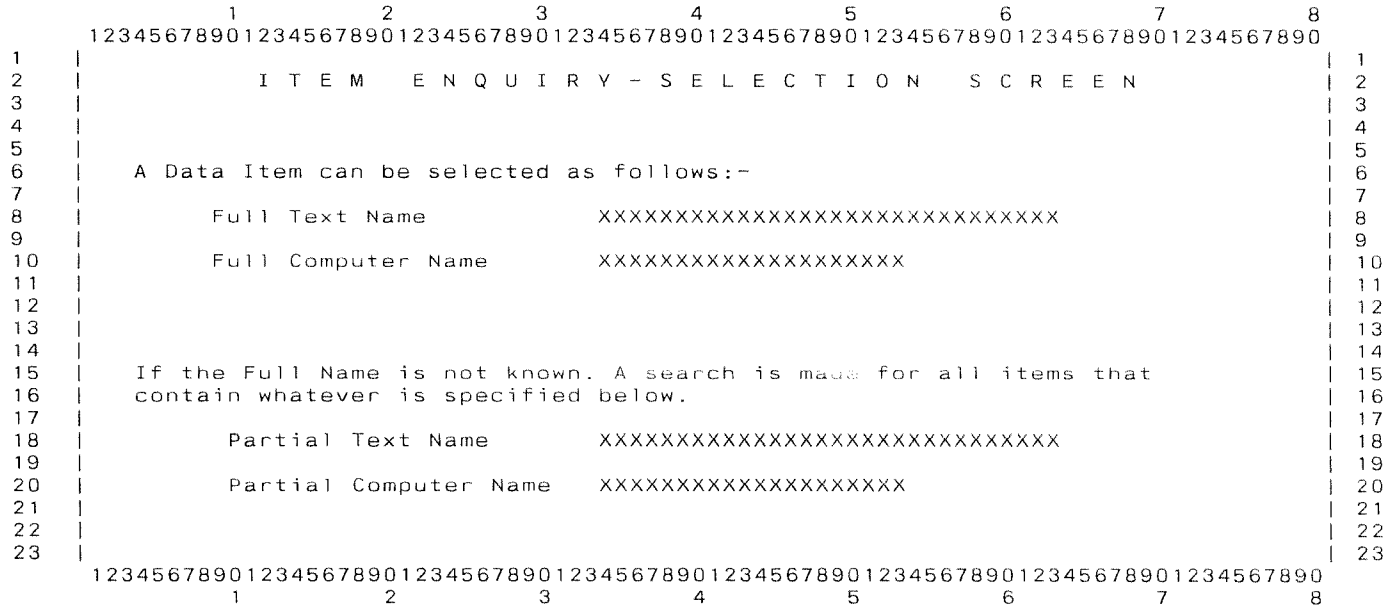

1 2 3 4 5 6 7 8
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
ITEM ENQUIRY - SELECTION SCREEN
A Data Item can be selected as follows:-
Full Text Name XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Full Computer Name XXXXXXXXXXXXXXXXXX
If the Full Name is not known. A search is made for all items that
contain whatever is specified below.
Partial Text Name XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Partial Computer Name XXXXXXXXXXXXXXXXXX

Figure 5.10

5.6 CODE MAINTENANCE

It is very rare for a system not to have any codes, and this system is no exception. Codes are a very useful means of quantifying data, and two types of codes are used in this system. They are:-

* Attribute Type Codes
* Computer Name Codes

5.6.1 Attribute Type Code Definition

The attribute type code is used to specify the different types of values that an attributes can take. This then allows reports to be produced which identify all attributes of a certain type, as per figure 5.9. e.g. It is then possible to get a list of all those fields that hold say a percentage value.

Data entry is performed using the screen shown in figure 5.11, and since all of the data entry is in a scrolled area, there is no limit to the number of codes that may be specified. This then gives the system greater flexibility so that it can be tailored to the individuals requirement. This ability to specify ones own codes seemed to be missing from a number of the Data Dictionaries, and it was certainly not available in the design systems, which is

unfortunate because the reports that are produced from these codes are invaluable.

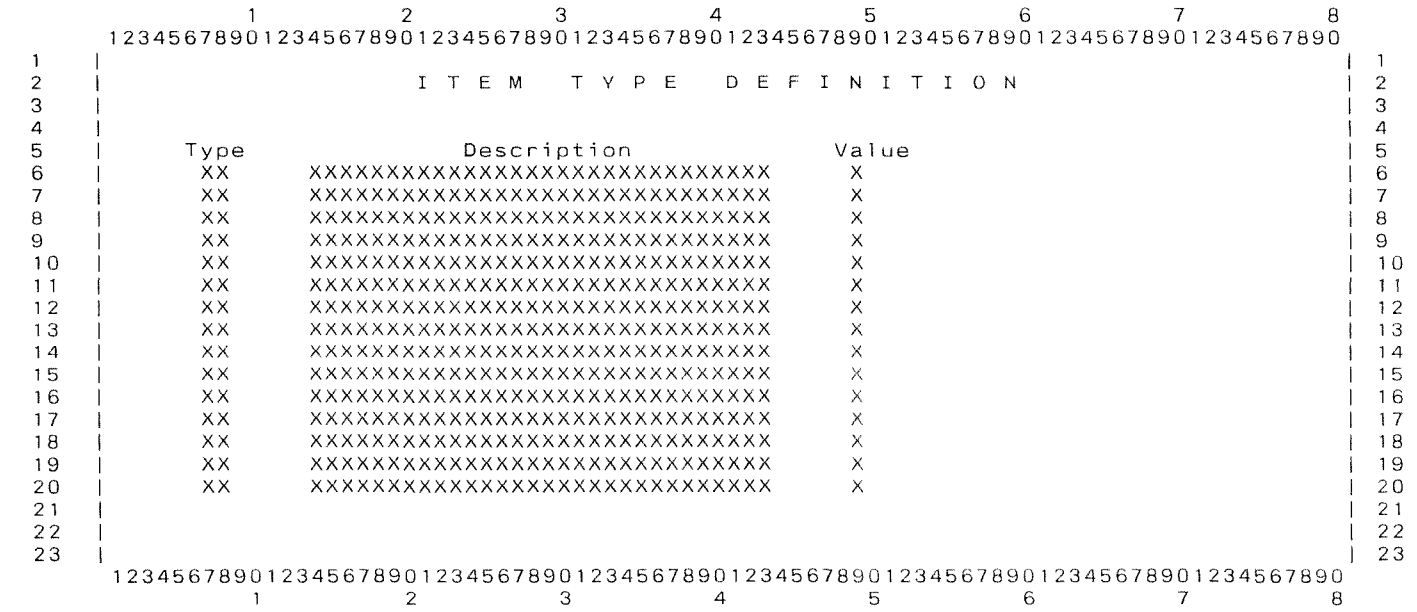

ITEM TYPE DEFINITION
Type
Description
Value
XX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X

Figure 5.11

5.6.2 Computer Name Codes Definition

The other code type that has proved very useful, but as of yet has no use within the actual design of the system is the computer name code. When an attribute name is translated to its computer name, certain conventions are often used for the name translation. e.g. Amount is always converted to AMT. By adopting a standard such as this, application programmers are then able to identify fields quickly, and avoid confusion over assumptions as to what a field may contain, because its name is ambiguous. Therefore it was decided to provide a facility where these codes could be defined, using the screen shown in figure 5.12.

The options available are:-

* Insert
* Amend
* Delete
* Print
    - o By Code (Refer Figure 5.13)
    - o By Description
* Display
    - o Code
    - o Part of the Code

- Part of the Description

In the installation where this system was developed, this facility has proved invaluable. Using this friendly system, the naming conventions can be established quickly, and they are then available for everyone to use. It was surprising that none of the Data Dictionaries provided a similar facility, because it is information pertinent to the model, and very necessary if standards are to be enforced. Keeping this information up to-date is far easier then a book, and then the information is available to everyone. In the future it is hoped to use these codes to derive and check the computer name from the attribute name.

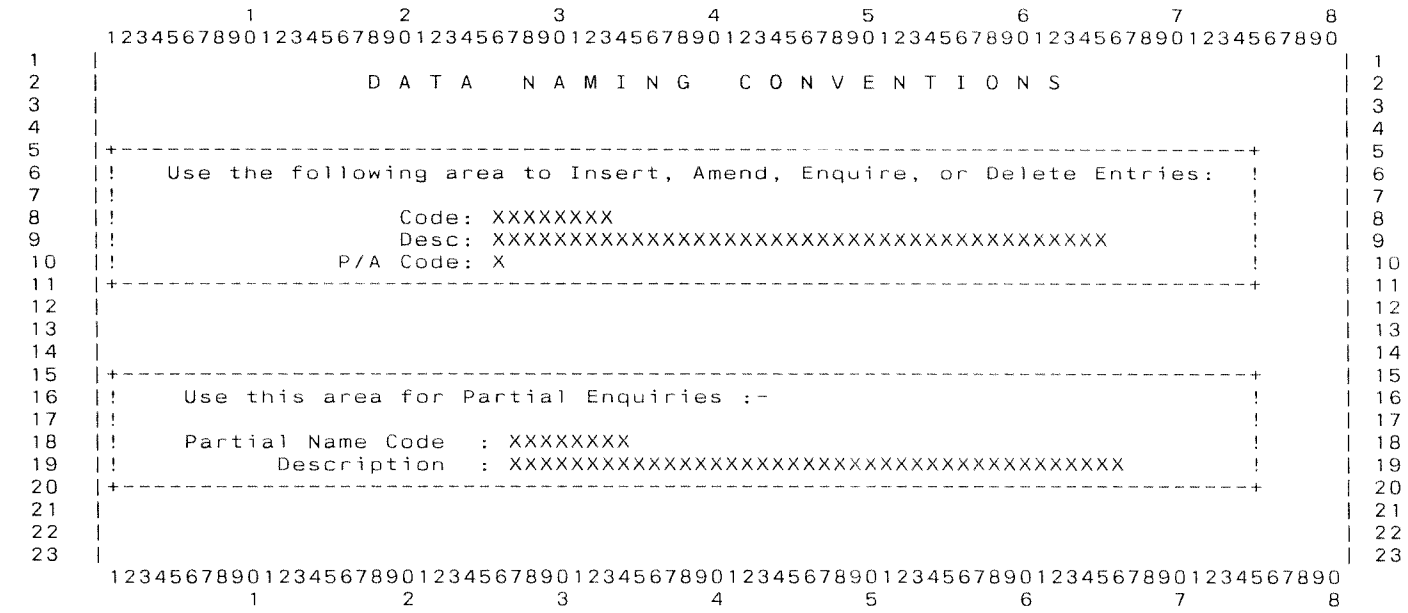

1 2 3 4 5 6 7 8
1234567890123456789012345678901234567890123456789012345678901234567890
DATA NAMING CONVENTIONS
Use the following area to Insert, Amend, Enquire, or Delete Entries:
Code: XXXXXXXX
Desc: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
P/A Code: X
Use this area for Partial Enquiries :-
Partial Name Code : XXXXXXXX
Description : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Figure 5.12

CODE	DESCRIPTION	PRIMARY OR ALTERNATIVE CODE
ABB	Abbreviation	A
ABBREV	Abbreviation	P
ACC	Account	A
ACCT	Account	P
ACCUM	Accumulation	P
ADD	Address	P
ADDR	Address	A
ADJ	Adjustment	P
AGE	Age Admitted	P
AGENT	Agent	P
AGT	Agent	A
ALLOC	Allocation	P
ANNIV	Anniversary	P
AP	Annualised Premium	P
ARR	Arrears	P
ASS	Assured	P
ASSEE	Assignee	P
ASSMNT	Assignment	P
BAL	Balance	P
BANK	Bank	A
BCH	Branch	P
BEF	Before	P
BEN	Benefit	P
BF	Brought Forward	P
BID	Bid	P
BK	Bank	P

Figure 5.13

5.7 SUMMARY

Upon successful completion of this part of the system, Entity Modelling will be complete. However, unlike other systems, this one has forced the source of every attribute to be defined, and in addition, comprehensive information about each attribute may be available. A weakness of the systems reviewed was that were extremely good at defining the attributes, but had no interest in retaining where the attribute came from, they also lacked the ability to report any errors that had occurred previously, This weakness has been overcome by the inclusion of the Document Error report.

Another useful by-product of this system is a simple, but powerful Data Dictionary. Although it lacks some of the facilities of the large systems, the tools available are more than adequate for the analysts and programmers who will be using the system.

Now that the Entity model has been constructed, we must now look at validating this model before the physical design may be contemplated.

Attribute Text Name
Attribute Computer Name
Mandatory Attribute Indicator
Derived Attribute Indicator
Type of Field
Picture Format
Occurs value
Primary/Candidate Key Identifier
Position in Key
Entity that this Attribute belongs to
Responsibility for Maintenance

Table 5-A

Attribute Text Name
Attribute Computer Name (Many may be defined)
Description of the attribute
Picture Format
Left or Right Justified
Occurs Value
Attribute Type Code
Status of Attribute i.e. Test,Production
Derived Attribute Indicator
Primary or Candidate Key Identifier
Position in Key
Default Values
Range Check rules or the name of an External Procedure which contains the validation rules
Entity that this attribute belongs to
Security Level required to read
Security Level required to update
Table used by application for conversion
Responsibility for maintenance
What creates this Attribute
Synoynms
Data Protection Act Classification
Date Attribute Inserted
Date Attribute Last Amended

Table 5-B

CHAPTER 6

TRANSACTION VERIFICATION

6.1 INTRODUCTION

Once the task of Entity Modelling is complete, the Entity Model must then be verified to check that the proposed model can handle the tasks expected of it. To verify the model, the Data Analyst must first identify all of the transactions that will be applied against the database. Each transaction is then processed through the model, and an iterative process of amending the model and reapplying the transaction is continued, until the transaction can be successfully processed. Not until all of the transactions can be processed by the model, will the designer be satisfied that the proposed model is suitable.

This verification of the model is very important, but it seemed to be the point when nearly all of the systems reviewed failed to provide a satisfactory solution to the problem. Some of the data dictionaries allowed a transaction to be defined using a rigid syntax, but once this information had been entered not one of

these systems attempted to validate this transaction against the information already in the dictionary. Instead all of the checking had to be performed manually which makes it potentially error prone, and could easily result in a transaction being overlooked.

What this system enables the user to do, is to describe the transaction in English text, rather than using a rigid syntax. Then both analysts and application programmers can refer to the same definition. Using the information already in the dictionary, the system will attempt to validate the transaction against the model, and then generate a simple diagram that illustrates the flow of the transaction through the model. To achieve this, a simple natural language processor had to be developed, which is a feature that is unique to this system.

Natural language is defined as any language that humans learn from their environment and use to communicate with each other. Until recently, all communication that humans have with computers has meant using a very rigid language that the computer understands without any ambiguities arising. Often humans find this rigid syntax very restricting, because it involves them having to express themselves logically. Therefore for some time the ability for humans to speak to computers using words and phrases that occur in everyday language has been a much sought after facility.

One of the reasons that this area has not been developed until recently is due to the complexity of the subject. The writing and understanding of a language involves constructing sentences.

The grammatical arrangement of words which shows their connection and relation, is referred to as syntax, but a sentence is only complete if it makes sense e.g. the sentence 'The chair ate the dog' is grammatically correct but does not make sense because chairs cannot eat. Therefore linguistics which is the study of languages and their meanings must be applied to each sentence to see if it is reasonable. Because the subject is so complex, humans often do not realise why they understand language, imagine the problems involved in trying to explain the process to a computer. Therefore most natural language systems set up a dialogue with the user, to question the information it is given, in an attempt to resolve the ambiguities that arise in the English language.

Recently some successful natural language systems have been developed and the success in this area is largely due to the application of programming languages such as LISP and the development of computers that are no longer designed for solving numerical problems. New products such as INTELLECT [INTEL] which enable users to interrogate a database using a natural language are already being heralded as the new 'fifth generation' languages. With the advent of machines that have separate processors for checking syntax, semantics, processing phrases of language, and hardware for analyzing speech, we should see rapid growth in this area in the coming years. [HARRIS] [LECHNER]

What this system attempts to show is that using only the languages COBOL and BASIC, a simple natural language processor can be built to interpret the transaction detail. Although one must not

expect miracles from this part of the system, it will be seen that this simple processor is able to cope with moderately complex transactions and it is an area that can be enhanced in the future. The use of the languages COBOL and BASIC was chosen because these were the only languages available in the commercial data processing department that was sponsoring the research. The program was predominately written in COBOL, and BASIC was only used for the string manipulation. It should be stated at the outset that this system does not attempt to establish a dialogue with the user, unlike conventional natural language processors. Instead it validates, reports anything that it cannot understand and waits for another validation request. Although this may be considered a rather unfriendly approach, it was decided that due to the volume of data that has to be searched, the response times may frustrate the user and waste their valuable time. Therefore the analyst specifies all of the transactions to be validated and waits for the results to appear. Using the report shown in figure 6.10 the results can be perused at the analysts leisure, and then the appropriate corrections can be made to the model.

6.2 TRANSACTION DEFINITION

Each transaction that the database must process is first defined using this facility. The transaction is given a reference, which is a unique code used to identify the transaction and then the following basic information is supplied for each transaction.

* Full Transaction Name
* Number of Transactions Expected in One Year
* Frequency of Transaction in Days

Once this information has been supplied the actual detail of the transaction may be specified in the free format area, which is a scrolled area, as shown in figure 6.1. For each transaction a record is also kept of when it was last validated and the outcome of that validation, all these details are then retained in the Data Dictionary in the record called TRANS-REC, refer figure 3.1. At this point no validation of the information entered takes place, because this is performed as a separate task. If any amendments are made to a previously validated transaction, then the indicator that states whether the validation has been completed is unset.

Within this facility the following options are available:-

* Insert
* Amend
* Enquire
* Delete
* Print
    - o All Transaction Details
    - o Individual Transaction Details
    - o All Transaction Verification Results
    - o Individual Verification Results
* Request to Verify a Transaction
* See the result of a Verification

All of the basic maintenance options are available. As stated previously the actual verification of the model is performed in a separate program to avoid unnecessary delays and prevent wasting the designers valuable time. Therefore when the user requests a transaction to be validated using the 'Request to Verify' option, this creates an entry on the file containing a list of all the transactions to be verified. When the verification is complete the outcome of the verification can be seen using the 'See the results of the Verify' option. For further details on transaction verification, please refer to the next section.

Several reports are also available within this facility. Either the transaction details may be reported, see figure 6.2 or the results of the verification, figure 6.3

This option has been designed so that if required, the data entry can be performed by a non-computer person because all that is required, is the hand written details from the data analyst.

```
             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
 1  |                                                                                | 1
 2  |                    T R A N S A C T I O N   D E T A I L S                       | 2
 3  |                                                                                | 3
 4  |      Trans Ref: XXXXXX                                                         | 4
 5  |          Desc:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                       | 5
 6  |                                                                                | 6
 7  |        Volume:  99999999        Frequency of Transaction in Days: 9999         | 7
 8  |Last Validated:  XXXXXXXXXXX                   Validation Complete: X            | 8
 9  |                                                                                | 9
10  |                                Transaction Details                             | 10
11  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 11
12  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 12
13  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 13
14  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 14
15  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 15
16  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 16
17  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 17
18  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 18
19  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 19
20  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 20
21  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 21
22  |X                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX              | 22
23  |                                                                                | 23
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
```

Figure 6.1

REFERENCE	SHORT DESCRIPTION	VALID	VOLUME	FREQ	LAST VALIDATED
ADM	Age Admitted The Client supplies evidence of their age and the Age Admitted Flag is updated.	Y	1500	1	13/09/86
ASS	Assignment Find the Client for the Policy and then create Assignee details, amend the Assignment Indicator and create the assignee address.	Y	450	2	13/09/86
CENQ	Client Enquiry Enquire upon the client details using either the client name or client reference.	Y		1	13/09/86
DD	Direct Debit On the Mandate Collection Day, create an arrear record. Then select the Premium Payment details and create a premium payment record. Finally update the unit record to reflect the value of the premium paid.	N	400000	7	14/09/86

Figure 6.2

REFERENCE	SHORT DESCRIPTION	VERIFICATION DETAILS
ASS	Assignment	| v S CLIENT | | v S POLICY | ASSIGNEE ----- END OF VERIFICATION ----
DD	Direct Debit	| v S POLICY *** UNKNOWN WORDS *** premium payment reflect value premium paid *** NO RELATIONSHIP FOR ENTITY UNIT ----- END OF VERIFICATION ----

Figure 6.3

6.3 TRANSACTION VERIFICATION

Once the user has already defined the transaction in the free format area, the problem now arises of trying to determine what the transaction does. To achieve this, each word in the description of the transaction, is identified separately and then using the principle of key word identification, each word is referenced to see if it is:-

* Attribute
* Computer Name
* Document
* Entity
* Other Word Type
* Synoynm

The information in the dictionary is the only source of reference that this system has. However, if everything has been defined correctly i.e. all the attributes and entities have been identified, then no other reference source is required.

Using this very simple technique, missing attributes or entities are quickly identified. During this searching of the dictionary the system can also identify multiple words such as 'Client Name' although the system would initially start searching

for the word 'Client' and then 'Name'. In this instance suppose the word 'Name' could not be identified. Then the previous word would be selected, forming the word 'Client Name'. A search would then be conducted on this basis. Once a word has been identified, searching for the next word is done on the basis of the type of the current word i.e. if the current word is an Entity, then the next word is most likely to be a miscellaneous type word.

As stated previously it was decided not to set up a dialogue with the user, instead the system reports all discrepancies on the error report, as shown in figure 6.10. These details are then retained in the verification area of the transaction, so that they can be displayed at any time.

The principle that this system uses is a very simple one, although it is recognised that it could be considerably enhanced. What the designer essentially needs to know is have any entities or attributes been omitted, and is the path taken by the transaction through the model correct. Therefore using the principle of key words, all of the words that cannot be identified are reported. Initially this may be a slow process, because additional words may need to be defined separately in the vocabulary extension section, please refer to section 6.4. Therefore the first verification of the transaction described in figure 6.4 could look something like the example in figure 6.5, where nearly every item is unknown. Of course this problem is easily overcome by trying to define all of these additional words before one starts. Once this has been done all subsequent verification should identify genuinely missing

attributes and entities, such as the example in figure 6.6. On this report one sees the words 'Broker' as unknown. Here the analyst knows that the system should know about a 'Broker', because it is an entity that should appear on the entity model. Therefore it looks like this entity has been overlooked, so the entity 'Broker' must be defined, before one may continue. Also the attribute 'First Premium' is unidentified. First the system is searched to see if the field has been named differently using the attribute enquiry facility. If the results of this search is negative, then a check is made to see that the document on which the attribute should appear has been defined. In this case, it is seen that the document 'Application' exists, but the attribute has been overlooked, so it is entered into the system and the verification is repeated. This time the verification is successful and the results are shown in figure 6.8.

Once the verification is complete, the system identifies each entity and attempts to determine the action to be applied to it, this information has been collected so that it can be used when the system is next enhanced. Once the system has identified all of the attributes and computer names that belong to an entity, using the relationships that have been defined on the entity model, a simple diagram is produced that shows the flow of the transaction through the model. It was decided to generate a diagram because this is a very effective method of describing the transaction in simple terms. Since the processing flow on a transaction only references some of the entities, drawing the diagram is not

as complex as drawing the entity model. Using the model as the reference point, if an entity is referenced and not reported in a relationship, or no starting point is defined , then the system will report an error as per figure 6.9, where a relationship has been defined between fund and unit, but no detail has been given on how to get to the fund information. A common fault when defining transactions is to omit the detail as to how one arrives at a certain entity, but this system forces this information to be supplied. Another example of this is the description of the transaction 'Application', which has been amended in figure 6.7 to reference the branch, which results in branch being featured in the flow diagram in figure 6.8.

The diagram that is generated (an example is shown in figure 6.8) will illustrate the transaction flow exactly as it is has been defined, no attempt will be made to reorganize the flow into a more efficient sequence. It was decided not to amend the sequence of the transaction so that the analyst can see from the diagram that although it may be a valid processing sequence it is not the most efficient route through the model.

After this simple diagram has been generated, the system will report any entities that do not have a relationship, along with all of the manual forms and records that have been specified as per the example in figure 6.8. Using this system, one soon discovers that although several attempts may be required to validate the transaction, sometimes it may be necessary to rephrase a sentence so that the system can understand it. e.g. For the transaction Client

Enquiry, the original definition was 'Change the Name of the Client'. The system had difficulty resolving this because it could not determine who 'name' belonged to. The transaction was rephrased to 'Change the Client Name' and the system instantly recognised Client Name as being an attribute of the Entity Client. Change was the action on the Entity and the resulting flow diagram was:-

The 'S' indicates that the entity Client is a starting point. This is one of the disadvantages of not setting up a dialogue with the user, because the problem could have been resolved by the system asking some questions.

It is hoped that users of the system will not find this too restricting, and since none of the systems reviewed attempted to validate the transactions, this feature is a bonus to the user. What has been shown is that it is feasible to describe the transaction in a natural language, without forcing the user to use a rigid syntax, and then validate this information against the details already held in the dictionary. This then gives the analyst some confidence that real checking of the model has been performed, and that nothing has been overlooked. It is very easy for someone to say, but of course the details regarding the collection of direct debits have been defined. Only to discover much later that a few key fields have been overlooked. Application programmers will not be very pleased if their schedule slips, because the database design

team have to amend the database because they omitted a few fields.

SCHRODER DATABASE DESIGN SYSTEM T R A N S A C T I O N D E T A I L S 10-OCT-1986 PAGE: 1

REFERENCE	SHORT DESCRIPTION	VALID	VOLUME	FREQ	LAST VALIDATED
APPLIC	Application The Broker submits an application that has been completed by the Client. This is allocated a policy number and then input into the system. The cheque for the first premium must be attached to the application, otherwise the application is rejected.	N	50000	300	00/00/00

Figure 6.4

SCHRODER DATABASE DESIGN SYSTEM T R A N S A C T I O N E R R O R R E P O R T 10-OCT-1986 PAGE: 1

REFERENCE	SHORT DESCRIPTION
APPLIC	Application

Broker	UNKNOWN WORD
submits	UNKNOWN WORD
that	UNKNOWN WORD
has	UNKNOWN WORD
been	UNKNOWN WORD
completed	UNKNOWN WORD
This	UNKNOWN WORD
allocated	UNKNOWN WORD
input	UNKNOWN WORD
into	UNKNOWN WORD
system	UNKNOWN WORD
cheque	UNKNOWN WORD
first	UNKNOWN WORD
premium	UNKNOWN WORD
must	UNKNOWN WORD
be	UNKNOWN WORD
attached	UNKNOWN WORD
otherwise	UNKNOWN WORD
rejected	UNKNOWN WORD
Application	MANUAL DOCUMENT

Figure 6.5

SCHRODER DATABASE DESIGN SYSTEM T R A N S A C T I O N E R R O R R E P O R T 10-OCT-1986 PAGE: 1

REFERENCE	SHORT DESCRIPTION
APPLIC	Application

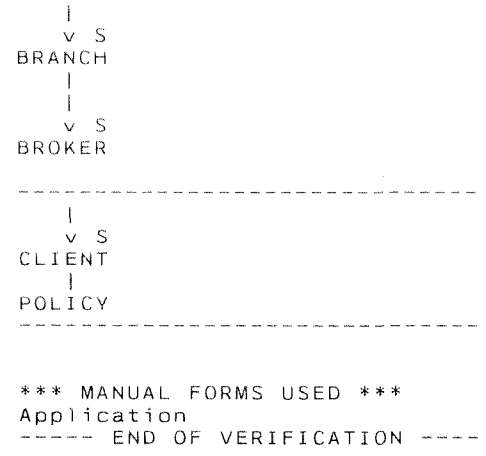
Broker	UNKNOWN WORD
submits	UNKNOWN WORD
allocated	UNKNOWN WORD
input	UNKNOWN WORD
system	UNKNOWN WORD
cheque	UNKNOWN WORD
first	UNKNOWN WORD
premium	UNKNOWN WORD
attached	UNKNOWN WORD
otherwise	UNKNOWN WORD
rejected	UNKNOWN WORD
Application	MANUAL DOCUMENT

Figure 6.6

REFERENCE	SHORT DESCRIPTION	VALID	VOLUME	FREQ	LAST VALIDATED
APPLIC	Application The Broker submits an application to the branch completed by the Client. This is allocated a policy number and then input into the system. The cheque for the first premium must be attached to the application, otherwise the application is rejected.	N	50000	300	21/10/86

Figure 6.7

APPLIC Application



*** MANUAL FORMS USED ***
Application
----- END OF VERIFICATION ----

Figure 6.8

REFERENCE	SHORT DESCRIPTION	VERIFICATION DETAILS
DD	Direct Debit	

```
    |
    v S
POLICY
    |
ARREAR

--------------------------------
FUND
    |
UNIT

NO STARTING POINTS
--------------------------------

------ END OF VERIFICATION ----
```

Figure 6.9

SCHRODER DATABASE DESIGN SYSTEM T R A N S A C T I O N E R R O R R E P O R T 13-SEP-1986 PAGE: 1

REFERENCE	SHORT DESCRIPTION		
DD	Direct Debit		
		Mandate	UNKNOWN WORD
		Collection	UNKNOWN WORD
		Day	UNKNOWN WORD
		arrears record	UNKNOWN WORD
		Premium	UNKNOWN WORD
		Payment	UNKNOWN WORD
		premium	UNKNOWN WORD
		payment record	UNKNOWN WORD
		unit record	UNKNOWN WORD
		reflect	UNKNOWN WORD
		value	UNKNOWN WORD
		premium	UNKNOWN WORD
		paid	UNKNOWN WORD
	NO STARTING POINTS IN TRANSACTION		
ASS	Assignment	VERIFICATION COMPLETED OK	
NAME	Name Change		
		Name	UNKNOWN WORD

Figure 6.10

6.4 VOCABULARY EXTENSION

If this simple natural language processor is to be successful then there are many words that the system will not understand, therefore its vocabulary must be extended. The system acknowledges three types of word although initially not all of the types are used.

* Key Word
* Process Word
* Miscellaneous Word

A Key word is defined as any action such as 'Create' or 'Delete'. A Process word indicates that some type of processing is about to occur e.g. For, While, Do . A Miscellaneous word, is one that is required to complete the sentence to create correct English i.e. 'the' or 'a'. All of these additional words are contained in the record KEY-WORDS-REC, refer figure 3.1. To assist processing, when a word is defined, one can also specify what type of word will most likely follow this word. e.g. Following a miscellaneous word, the next word will probably be an entity, an attribute or a computer name. There is no restriction on the number of words that the system may hold, but they must all be input manually by the users of

the system. Unlike a real natural language processing system that expands its own vocabulary.

Within this facility the following options are available:-

* Insert
* Amend
* Enquire
    - o Search on the whole word
    - o Search on part of the word
    - o Display all words of a given type
* Delete
* Print
    - o All words
    - o All of a certain type

All maintenance is performed via the screen shown in figure 6.11 and a sample report illustrating the type of words held in the dictionary is shown in figure 6.12.

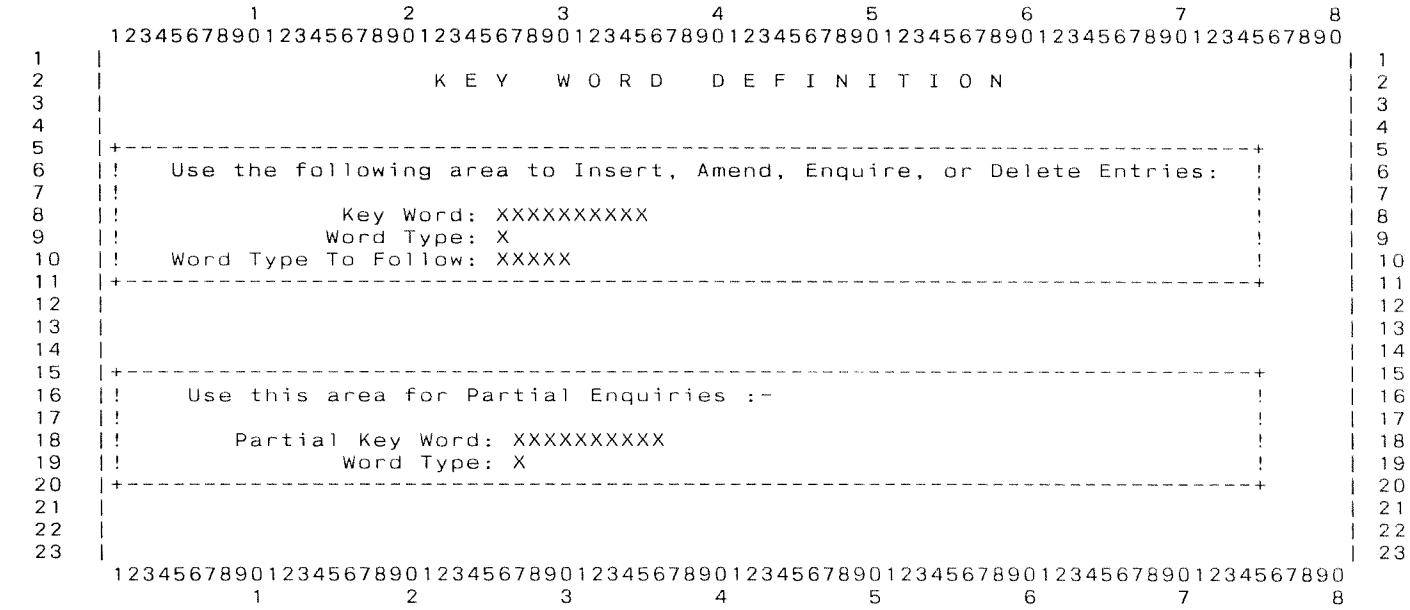

K E Y W O R D D E F I N I T I O N
Use the following area to Insert, Amend, Enquire, or Delete Entries:
Key Word: XXXXXXXXXX
Word Type: X
Word Type To Follow: XXXXX
Use this area for Partial Enquiries :-
Partial Key Word: XXXXXXXXXX
Word Type: X

Figure 6.11

SCHRODER DATABASE DESIGN SYSTEM K E Y W O R D D E F I N I T I O N 16-SEP-1986

KEY WORD	WORD TYPE	TYPE OF WORD TO FOLLOW
A	M	
ASSIGN	K	E
BY	M	
CHANGE	P	E
CREATE	K	P
DO	P	
EITHER	P	A
ENQUIRE	K	M
EVIDENCE	P	M
FIND	P	A
INSERT	K	A
IS	M	
OF	M	
ON	M	
OR	P	

Figure 6.12

6.5 ENTITY/TRANSACTION REPORTS

Once the verification is complete two reports are available which detail:-

* All the Entities Used on a Transaction - Refer Figure 6.13
* For a given Entity, the transactions where it is used - Refer Figure 6.14

SCHRODER DATABASE DESIGN SYSTEM E N T I T Y / T R A N S A C T I O N U S A G E R E P O R T 21-MAR-1987 PAGE:

ENTITY	TRANSACTION
	Age Admitted
CLIENT	
	Agent Enquiry
AGENT	
	Application
BROKER	
BRANCH	
CLIENT	
POLICY	
	Dummy Transaction
	NO ENTITIES REFERENCED FOR THIS TRANSACTION
	Name Change
CLIENT	

Figure 6.13

ENTITY	TRANSACTION
AGENT	Agent Enquiry
ARREAR	Direct Debit
ASSIGNEE	Assignment
BENEFIT	Direct Debit
CLIENT	Age Admitted Application Assignment Client Enquiry Name Change
CLIENT-STUB	Assignment
DOCTOR	NO TRANSACTIONS USE THIS ENTITY
FUND	Direct Debit
MEDICAL REPORT	NO TRANSACTIONS USE THIS ENTITY
PERIODIC PREMIUM	Direct Debit
POLICY	Application Assignment Direct Debit

Figure 6.14

6.6 SUMMARY

The intricate detail of natural language processing has been deliberately omitted from this chapter because this system does not pretend to be a true natural langauage processor, consequently its processing is not as complex. It is an area of the system that it is recognised could be considerably enhanced, not only in its understanding of the transaction, but also in the presentation of the diagram illustrating the flow through the model. Its weaknesses are acknowledged, but the fact that this system provides a function that no other system does, and that the results are very satisfactory, this initial release illustrates that the pre-requisites have been met. At least once this stage has been completed, the designer can rest assured that no attributes or records are missing from the database, provided of course, all of the transactions have been defined. Now it is possible to proceed into normalisation knowing that the construction of the physical schema is very near and it should no longer be necessary to amend the model further during this implementation.

CHAPTER 7

NORMALISATION

7.1 NORMALIZING

Each part of the system that has been discussed previously, has always required a minimum quantity of data entry to be performed. That is, the Entity Model had to be defined, then the documents and finally the transactions. The next step to be reached in database design is normalisation, which in this system is optional, because it is not a pre-requisite for a CODASYL database.

The concept of normalisation was first introduced by Codd for use with the relational data model. Its purpose is to ensure that a relation contains only those attributes which depend on its key, where 'depend on' means that each value of the key identifies just one value of the attribute. The technique was initially devised for constructing relational databases, but it can be applied to the network model, if required. Normalisation is optional in this system because there are drawbacks in normalizing a CODASYL database. One of the main disadvantages is the performance problems

that arise from normalizing. For example, one of the first steps is to remove all repeating groups into a separate record. Therefore imagine an order that can have five lines of detail. In a normalized database the resulting I/O would be six reads, one read for the order details and five reads for each of the lines. Unnormalized, only one I/O would be required to read the order complete with all of its line detail. e.g. the order record would contain all the lines of the order. Another problem with normalisation is that it eliminates duplicate data item, but this may be desirable to reduce system I/O or maintain integrity of the data e.g. In an insurance database, information is retained about policies and their benefits. These are two separate records, but each benefit record for the policy, contains the policy number to ensure that the benefit record is attached to the correct policy. A CODASYL database allows a 'CHECK' clause to be supplied which performs this check automatically, consequently if the data is normalized this check cannot be performed. Consequently the designer must assess the type of processing that the system will do and take performance problems like these into consideration.

Codd initially defined three levels of normalisation first, second and third normal form, later this was enhanced to include fourth and fifth normal form, but only the first three types will be discussed here. First Normal Form is defined as when an attribute does not have any multiple values for a given primary key. Second Normal Form is defined as when first normal form is complete, and every non-key attribute is fully functionally dependant on the key.

Third Normal Form is defined as when second normal form is complete, and there are no transitive dependencies. Each of these stages is best explained by means of an example taken from an ordering system, the details of which are outlined in figure 7.1, where Order Reference Number is the only key.

ORDER

Order Reference Number (Key)

Customer Number

Customer Name

Order Receipt Date

Product Number

Product Name

Order Quantity

Order Value

Total Order Value

Figure 7.1

To complete first normal form, remove all of the attributes that have multiple values for a given key. In the example this would be the product order details, because for a given order, several products may be requested. Therefore after First Normal Form the record would be as per figure 7.2

ORDER

Order Reference Number (Key)

Customer Number

Customer Name

Order Receipt Date

Total Order Value

ORDER ITEM

Order Reference Number (Key)

Product Number (Key)

Product Name

Order Quantity

Order Value

Figure 7.2

Second normal form involves removing all of the attributes that are not dependent on the whole of the multiple key. In the example shown in figure 7.2, in the Order-Item record, the attribute Product Name is dependent only on the part of the key Product Number, not on the other part of the key which is Order Reference Number. Therefore Product Name has to be removed from the Order-Item record and a new record is created as per figure 7.3.

ORDER

Order Reference Number (Key)

Customer Number

Customer Name

Order Received Date

ORDER ITEM

Order Reference Number (Key)

Product Number (Key)

Order Quantity

Order Value

PRODUCT

Product Number (Key)

Product Name

Figure 7.3

Third Normal Form involves removing all of the attributes whose values depend on the values of other attributes which are not part of the key. In figure 7.3 the attribute Customer Name is dependant on the value of the the attribute Customer Number. However, Customer Number does not form part of the key, therefore it should be removed and put into a separate record called CUSTOMER, which contains only Customer Number and Name. The outcome of third normal form is shown figure 7.4

ORDER

Order Reference Number (Key)

Customer Number

Order Received Date

ORDER ITEM

Order Reference Number (Key)

Product Number (Key)

Order Quantity

Order Value

PRODUCT

Product Number (Key)

Product Name

CUSTOMER

Customer Number (Key)

Customer Name

Figure 7.4

This system will perform First Normal Form analysis, by taking every attribute of the entity that is specified as an occurs value and creating a new record that is called the entity name plus stub. e.g. the attribute address belongs to the entity Client and it occurs 5 times. Therefore a entity is created called CLIENT-STUB which contains only the attribute address, and a 1:M relationship between the entities CLIENT and CLIENT-STUB is created. Therefore if the client had six addresses, then for one client record there

would be six client-stub records. One can already see that trying to read all of the addresses for a client would involve considerable I/O. That is why this system does not enforce normalisation, because the grouping of attributes together in an occurs may be very desirable to improve overall system performance. Although this system does not perform second normal form analysis, the technique that is used for allocating attributes to an entity, means that second normal form is being performed indirectly. For example this system would automatically allocate the attribute Product Name to the entity Product, not the entity Order.

As can be seen, performing third normal form analysis is a complex task, which requires an understanding about the dependencies between the attributes and their keys. To determine this information is no easy task and since it was decided that performance problems can result from normalizing databases, it was decided to restrict the extent to which the data is normalized.

To normalize the data, the user must first specify whether normalisation is required using the System Control Maintenance Option described in section 7.2. Then the normalisation option is selected from the menu, the data is normalized, and a report is produced as per figure 7.5. [DATE] [DATIBM]

SCHRODER DATABASE DESIGN SYSTEM N O R M A L I S A T I O N R E P O R T 20-SEP-1986 PAGE: 1

Normalizing Attribute Address

Created New Entity CLIENT-STUB

No Attributes Allocated to Entity PRODUCT-STUB Deleted Entity

Figure 7.5

7.2 SYSTEM CONTROL MAINTENANCE

Various system control information is needed at this stage of development and this facility enables the user to specify the control information. A screen is shown as per figure 7.6 which details:-

* If any changes have been made to an occurs field and the date on which this occurred
* Date Action Report last generated and if it completed ok
* Whether all transaction have been validated and date last validation occurred
* Date on which normalisation was last attempted
* Date Schema was last generated and number of errors

The following information may then be specified by the user:-

* Whether Range Checks are to be included in the Schema
* Normalisation required
* Schema Name
* Default names for Area
* Suffixes for

- o Area Name
- o Record Name
- o Set Name

* Default Order Clause
* Default Insertion Clause
* Default Retention Clause

It is this facility that is used to specify if normalisation is required by simply setting the indicator to Y if it is required, and N if it is not.

Once normalisation has been completed, then the final part of the database design may be attempted, that of schema creation.

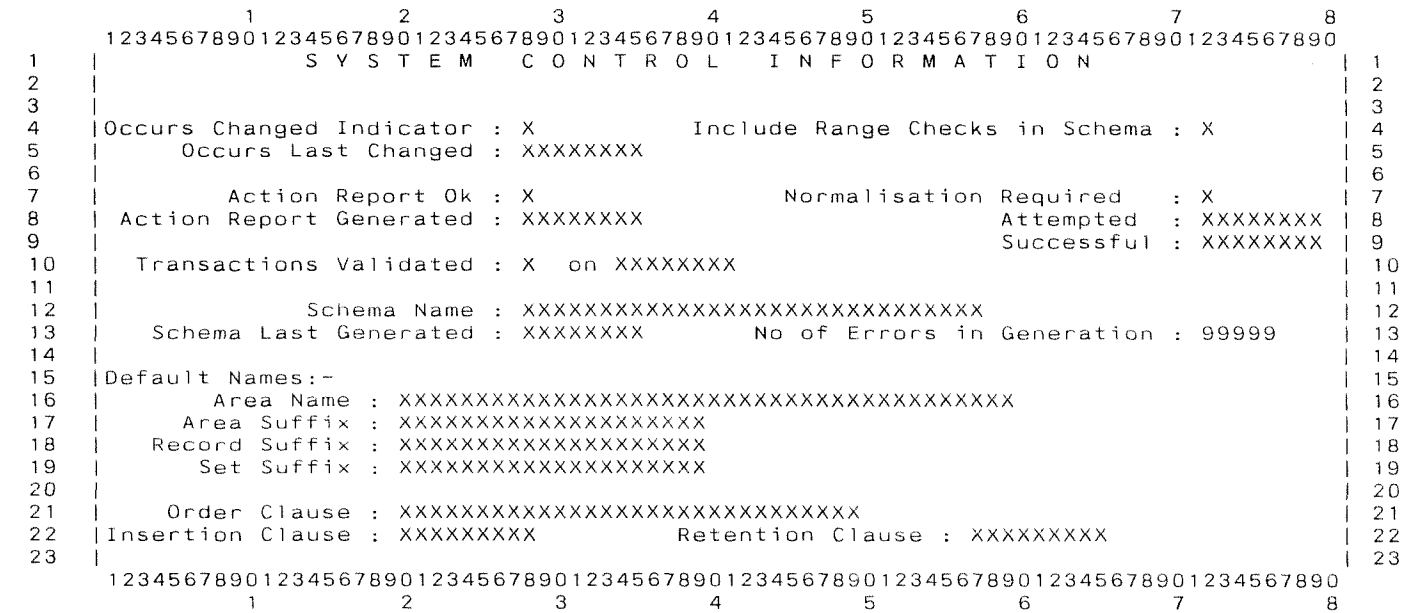

SYSTEM CONTROL INFORMATION
Occurs Changed Indicator : X
Include Range Checks in Schema : X
Occurs Last Changed : XXXXXXXX
Action Report Ok : X
Normalisation Required : X
Action Report Generated : XXXXXXXX
Attempted : XXXXXXXX
Successful : XXXXXXXX
Transactions Validated : X on XXXXXXXX
Schema Name : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Schema Last Generated : XXXXXXXX
No of Errors in Generation : 99999
Default Names:-
Area Name : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Area Suffix : XXXXXXXXXXXXXXXXXXX
Record Suffix : XXXXXXXXXXXXXXXXXXX
Set Suffix : XXXXXXXXXXXXXXXXXXX
Order Clause : XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Insertion Clause : XXXXXXXXX
Retention Clause : XXXXXXXXX

Figure 7.6

CHAPTER 8

SCHEMA CREATION

8.1 INTRODUCTION

Now that all the previous steps have been successfully completed, it is possible to create the schemas for our CODASYL based database. The schema describes the database that is to be created, and in a CODASYL compliant database, three types of schema have to be defined:-

* Master Schema
* Storage Schema
* Subschema

The master schema is the definition of the database which has no regard for the physical implementation, which is described by the storage schema. Then there are a number of subschemas, which are used by the applications to access the database.

One of the advantages of using a DBMS is that an application need only view those records and attributes that it needs for its

processing. To achieve this, the view of the data that the application sees is controlled by the subschema, which consequently results in a number of subschemas being created.

Until this stage, all the results of the work that has been done previously, has been entirely independent of any machine. That is the entity model is not concerned with any machine, it is a business description. The same applies to the documents, attributes and transactions which are also descriptions of the business. However, at this stage we are now trying to actually implement and create a schema, and this will have to be put onto a particular machine.

Although it was stated at the outset, that this system would create a CODASYL compliant schema, like all other standards that are recommended in the industry, computer manufacturers never implement them as originally specified, and the CODASYL standards are no exception. The CODASYL standards have evolved from the Database Task Group that was set up in 1965. Originally this group was known as the List Processing Task Force, but this name was thought to be unsuitable and was soon changed. Late 1969 the group published its first set of specifications and finally in 1971 its recommendations were accepted. It was at this time that the Data Description Language Committee was formed, and it is now this body that recommends the changes to the standards. A number of database systems are available that are based on these standards, they include Univac's DMS 1000, Cullinane's IDMS and Digital's VAX-11 DBMS. Although these systems are all CODASYL compliant, the

manufacturers often choose to omit certain functions, or implement them slightly differently from the recommendations. The prototype system developed for this thesis, is bsed on Digital's VAX-11 DBMS, which is no exception to this rule. Therefore the schemas generated by this prototype system will only be suitable for the VAX-11 database management system. Consequently to convert this system to run on another machine, some changes would be required to the programs that generate the schemas, to amend the format to that manufacturers syntax. Despite this problem, the purpose of this thesis is to show that it is possible to create the schemas, a facility that as was shown in chapter two, none of the systems reviewed provided.

8.2 SCHEMA CREATION

Before the schema may be constructed, all of the previous stages of design must have been successfully completed. Therefore the Document Action Report must contain no errors, this will ensure that all the attributes are allocated to an entity. All of the transactions must have been validated against the model, and if normalisation is required, this must also have been completed. If any of these steps has not been completely successfully, an error will be generated because an invalid schema may result.

None of the systems reviewed attempted to create the physical schema. Using the entity model as the basis for the physical implementation, which is one of the common techniques used

in database design. [JUKES] The system developed here follows the following rules to translate the conceptual model to the physical implementation.

1. Each entity on the entity model that has attributes becomes a record
2. Each relationship becomes a set provided the relationship is not Many-to-many
3. If an entity is a starting point, then this is created as an entry point into the system
4. For many-to-many relationships, a junction record and junction sets are created
5. The default area name is used for record placement unless otherwise specified

As described in Chapter 7, before the schema may be created, using the control maintenance option the schema name, default area, record and set suffixes, and insertion and retention clauses must be specified. Each schema will be given the name specified in the schema field in the control maintenance option. Then it will be allocated a version number and its date of creation will be todays date, therefore the system is not restricted to retaining only one version.

Some systems like to suffix all of their areas, records and sets with a standard suffix. e.g. an area is always

<name>_AREA

This system will automatically allocate the suffix, if it has been specified using the control maintenance option.

In a CODASYL database, the area equates to an actual file in conventional file terms, therefore there must be a minimum of one area, with no upper limit. What this system is unable to determine is how many areas are required. Ideally what one is trying to achieve is to group into one area all records that are used for similar processing. This results in improved performance because one I/O to the database will read a number of different record types into a buffer. which hopefully will be used during the transaction. To create one area per record would be very inefficient, because this would create excessive number of files and impact upon performance, by potentially creating a lot of files which could all be held on the same disk. Another alternative would be to group records together according to their name, but this is also not very satisfactory, because records may be grouped together that have nothing in common. What is really needed is to know how the transactions will be processed in the context of the entire system so that the areas can be correctly placed on the disk. This is a potential future enhancement and it was felt that in this first release, this aspect of the design would best be left to the designer to obtain the optimum performance from the database. Initially, the system creates one area, using the default area name, which contains the entities that have previously been specified. Additional areas can be specified by simply specifying the area name on the entity definition screen, refer figure 4.3, the entity will then be placed in this area in preference to the default area. Remember that each entity becomes a record, therefore to place the entity CLIENT in a separate area, the area name for the entity

CLIENT would be amended to CLIENT-AREA. Once the areas have been defined, the records are then created.

For each entity that has attributes specified, a record is created. This is the usual translation from the entity model to physical design and all the attributes for the entity are identified as items of the record. Now one can see an important use of entity modelling, not only does the model form the basis of the translation to the physical schema, but the attributes of each entity form the detail. A description is given to each item by using the text attribute name. Also included will be any default values that may have been specified. A sample output is shown in figure 8.1

```
RECORD NAME IS  CLIENT_REC
WITHIN CLIENT_AREA

   ITEM NAME IS  CLIENT_NM            TYPE IS CHARACTER 30
*  Client Name

   ITEM NAME IS  SEX                  TYPE IS CHARACTER 1
*  Sex

   ITEM NAME IS  DOB                  TYPE IS UNSIGNED NUMERIC 8
*  Date of Birth
```

Figure 8.1

One facility that this DBMS provides which is not common in CODASYL compliant databases is the inclusion of the check clause. Using the control maintenance screen one can specify if these clauses are to be included. If required, then the exact contents of the range check clause or external procedure will be included in the schema for the specified attribute. Therefore the user must ensure that the contents of the field complies with the schema syntax. It was decided to do it this way because then the flexibility was available

to allow the user to specify a range of values without being tied to a rigid format. Therefore, if the check clause for the field Status is to be included in the schema, then it would have to be specified in the range check field as

STATUS = "P" OR STATUS = "L"

where Status is the computer name. Alternatively if the check clauses were not to be included, then the range check field or external procedure could be used as a free format text area. Then the field would probably be defined as

P = Production

L = Live

In the CODASYL standards a check clause may also be specified between several items, or on a set. Currently this system caters for item check clauses only, but the inclusion of the other types is a potential future enhancement.

One of the problems with any model is the implementing of *many-to-many* relationships. This is a difficult structure to implement because insufficient pointers are available within the network to define this type of relationship. To overcome this, a common technique for resolving this problem is the creation of a *junction record between the two entities*. The system creates a new record named using the two entity names and containing the keys from each record. e.g. in an insurance application there is a m-m relationship between the the entities Client and Policy. The keys are Client Reference and Policy Number respectively. Therefore a new record is formed called

which contains the attributes Client Reference and Policy Number.

For each record created, and for each relationship in which the record participates a set is created using the following naming standards. If the entity is a starting point then the name is ALL- to denote an entry point and then the entity name. The insertion and retention clauses are always set to AUTOMATIC FIXED, which means that when the record is stored, it will automatically be inserted into the set, and it must remain a member of that set until it is deleted. It is always created as a CALC set, which provides a fast access with minimal overhead. If this entity has a primary and a candidate key, then the candidate key will be specified as a sorted index set, since the calc algorithm is only suitable for locating primary records. The candidate key set will be created as ALL-OTH to indicate it is an alternative access path. An example of this is shown in figure 8.2 where the entity Client has two entry points, one using client name and the other using client reference. Client name is the primary key so this is used for the ALL-CLIENT-SET which is a CALC set. Then for the candidate key, the set ALL-OTH-CLIENT-SET is created, which is an index set, sorted on client reference.

```
SET NAME IS  ALL_CLIENT_SET
OWNER IS   SYSTEM
MEMBER IS CLIENT_REC
INSERTION IS  AUTOMATIC
RETENTION IS  FIXED


SET NAME IS ALL_OTH_CLIENT_SET
OWNER IS   SYSTEM
MEMBER IS CLIENT_REC
INSERTION IS  AUTOMATIC
RETENTION IS  FIXED
ORDER IS SORTED BY ASCENDING
                                  CLIENT_REF
DUPLICATES ARE NOT ALLOWED
```

Figure 8.2

Duplicates are always specified as being not allowed because it was considered that sorted sets would probably be used as entry points to the system where duplicate key values might not be permitted. A future enhancement will allow one to specify whether duplicates are allowed and if so, in which order the duplicate records are to be placed i.e. first or last

Then for each relationship is which an entity participates a set is created called :-

<owner entity name> < _ > <member entity name> <set suffix>

Therefore the relationship between the fund and unit records would result in a set called FUND-UNIT-SET. These sets are all created with the default insertion, retention and order clauses specified on the control maintenance screen, a future enhancement will enable these to be specified specifically for the set. A typical example is shown in figure 8.3

```
SET NAME IS FUND_UNIT_SET
OWNER IS FUND_REC
MEMBER IS UNIT_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER IS LAST
```

Figure 8.3

The outcome of all these steps is the generation of a schema, an example of which is shown in figure 8.11. This schema is held within the system, and it may be extracted for amendment. The amended version may then be returned to this system provided the version number is increased. The system will not recognise any changes that have been made to this schema, so if it is re-created, it will be created as per the version prior to amendment, unless the model is amended for the changes that have been made.

8.2.1 Storage Schema

Once the schema has been created, the storage schema which represents the physical implementation may then be created by the system.

All records that do not provide an entry point into the system will be placed clustered by their owning record, which will provide a fast retrieval time. Records serving as entry points will be scattered using their keys, the scattering algorithm and the CALC algorithm will generate the same page on the database where the record should be located. For a record that participates in a number of sets, the first set encountered will be the one used for

placement. This may cause a problem and it is recognised that it is a part of the system that needs to be considerably enhanced. Any data item that is longer than 25 bytes will be given an 'ALLOCATION IS DYNAMIC' clause which causes the data item to be compressed in the database when it is stored. as per figure 8.4.

```
   ITEM NAME IS  CLIENT_NM                    TYPE IS CHARACTER 30
                                                ALLOCATION IS DYNAMIC
*  Client Name

   ITEM NAME IS  CLIENT_REF                   TYPE IS CHARACTER 8
*  Client Reference
```

Figure 8.4

For any set that has been created as a sorted index, the volumes for the entity, now a record, will be used to calculate the size of the index node. Otherwise the set will be specified as a chain set. Examples are given in figure 8.5

```
    SET NAME IS AGENT_POLICY_SET
    MODE IS INDEX
       NODE SIZE IS  207 BYTES

    SET NAME IS CLIENT_POLICY_SET
    MODE IS CHAIN
```

Figure 8.5

The outcome of all this is the generation of a storage schema, an example of which is shown in figure 8.12

8.2.2 Subschemas

A database may have a number of subschemas, the reason for this is that an application need not see the whole of the database, instead it is given a subschema which contains only those items and records that it needs to see. This is one of the methods used for

security of the data, because if the application does not have access to a particular field in the subschema, then it cannot see its contents, which is a feature that cannot be provided with conventional files.

This system will generate two types of subschema, either a default one that provides access to all data, or one that will handle only particular transactions. This is achieved by completing the screen shown in figure 8.8. If only specific transactions are to be included in the subschema then in this initial release, it was decided not to restrict the items in a record that were available. Therefore all the records referenced in the transaction description are included and although the system can derive from the transaction description that only certain fields are used, it was felt that in these early stages of design, it would be better to display all the items for a record, in case some part of the description has been omitted. Later an option could be included to specify whether mentioned fields were only to be included in the subschema. Now some form of security of the data is resulting and very important use is being made of the transaction descriptions that have been described previously. It is anticipated that if users know that information entered previously will be used later in the system, more care might be taken when the details are being supplied.

When the schema is generated a 'realm' has to be created to access the areas of the database. Normally several areas are grouped together into one realm, because it is the realm that is readied to determine how the database should be accessed. i.e.

concurrent or exclusive, read or update. This system is unable to determine how the areas should be grouped together, so it creates one realm per area. Unless the application specifically wants to access one realm differently, then the number of realms defined in the subschema is entirely transparent to the application. Each record is then defined, along with all of its attributes. Therefore the subschema begins as per figure 8.5

```
SUBSCHEMA SUBSCH FOR SDBC SCHEMA

REALM  ALL_AREAS_REALM                IS  ALL_AREAS
REALM  CLIENT_REALM                   IS  CLIENT_AREA

RECORD NAME IS  AGENT_REC

   ITEM NAME IS  AGT_NO                TYPE IS CHARACTER 8
*  Agent Number
```

Figure 8.5

Any group names that have been defined for an item will automatically be included in the subschema. The definition of group names is explained in the next section. A sample group is shown in figure 8.6

```
  GROUP NAME IS TOTAL_PRICES
  GROUP NAME IS PRICES
   ITEM NAME IS  BID_PRICE          TYPE IS UNSIGNED NUMERIC 9 -4
*  Bid Price

   ITEM NAME IS  OFF_PRICE          TYPE IS UNSIGNED NUMERIC 9 -4
*  Offer Price
  ENDGROUP  PRICES
  ENDGROUP  TOTAL_PRICES
```

Figure 8.6

The subschema only requires the names of the sets that will be used. Consequently only a list of set name has to generated As per figure 8.7

```
SET NAME IS UNIT_PRICE_SET
SET NAME IS OTH_UNIT_PRICE_SET
```

Figure 8.7

The system will hold any number of subschemas and as with the other schema types they may be extracted, edited and replaced in the system. An complete example of a subschema created by this system is shown in figure 8.13 An example of a schema for the Client Enquiry transaction is shown in figure 8.14

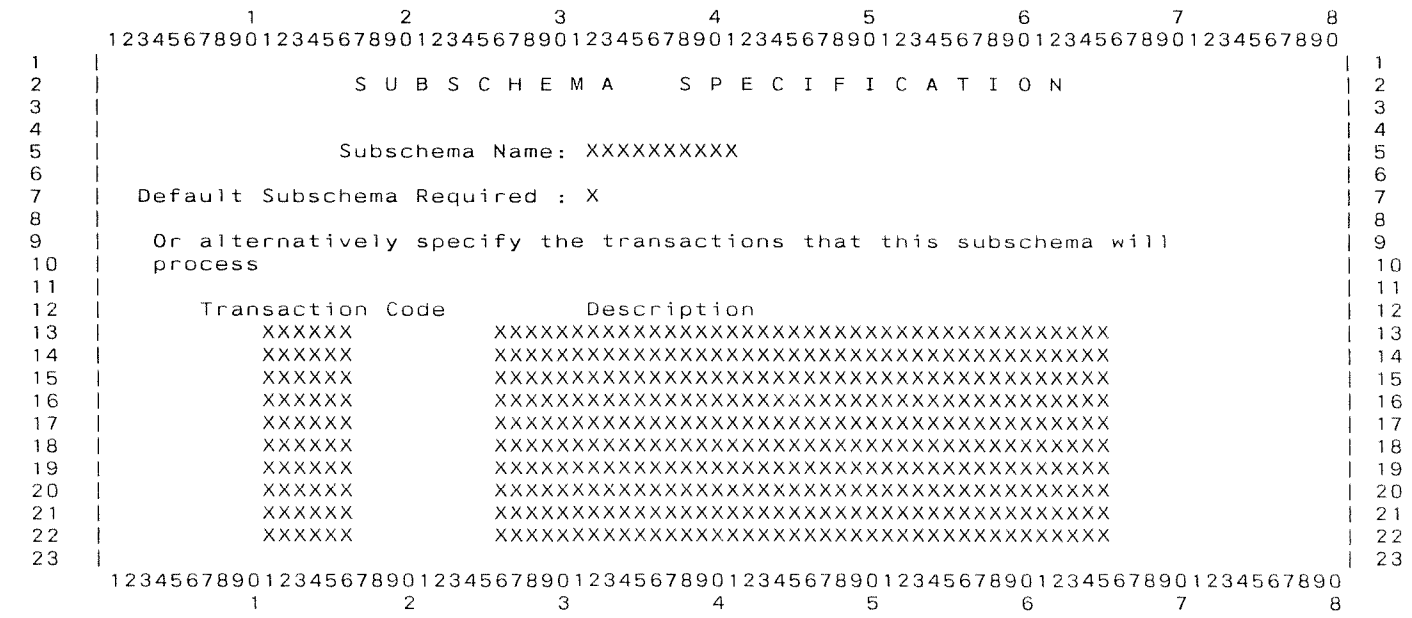

1 2 3 4 5 6 7 8
1234567890123456789012345678901234567890123456789012345678901234567890
S U B S C H E M A S P E C I F I C A T I O N
Subschema Name: XXXXXXXXXX
Default Subschema Required : X
Or alternatively specify the transactions that this subschema will process
Transaction Code
Description
XXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Figure 8.8

8.2.2.1 Group Names

Most computer systems need the ability to be able to define group names. A group name is a name used to reference a number of attributes as one attribute. An example of this is the group name CLIENT-NAME which is actually two attributes FORENAME and SURNAME. Therefore if the field forename had the contents LILIAN and the surname was HOBBS, then the attribute CLIENT-NAME would display as LILIANHOBBS. As one can see they are a very useful feature, but not all systems use them so they are optional within this system and are only included if defined. To define a group name the user is presented with a screen as shown in figure 8.9 A group name may either specify another group name or list some attributes that have been defined in the system. Therefore to maintain integrity of the system, group names are defined from the bottom up. Either the attribute text name or the computer name may be used. An optional report is also available which lists all the group names, a sample is shown in figure 8.10

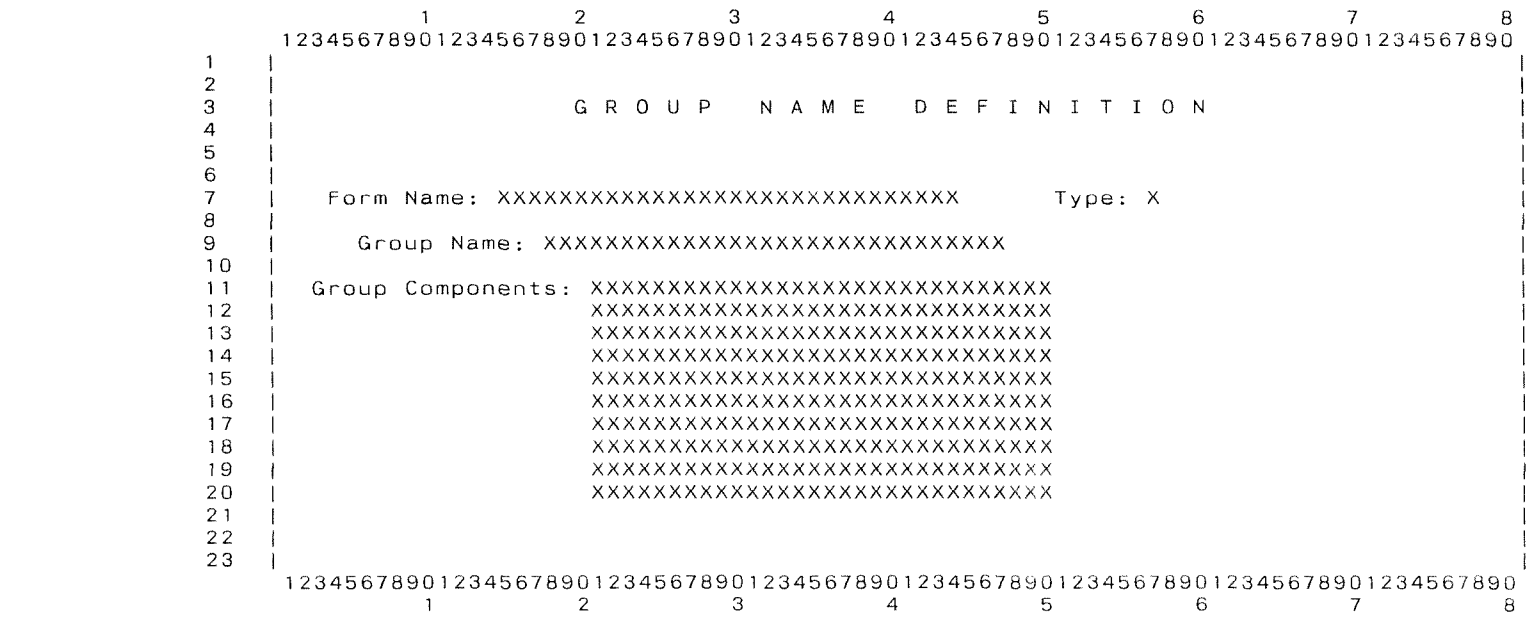

1 2 3 4 5 6 7 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
G R O U P N A M E D E F I N I T I O N
Form Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Type: X
Group Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Group Components: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Figure 8.9

FORM	DETAIL
Application	AGE DETAILS Age Age Admitted
Prices	TOTAL_PRICES PRICES Bid Price Offer Price

Figure 8.10

8.2.3 Summary

This chapter has shown that using all the information provided previously, it is possible to create a schema from the information held within this system. Admittedly, the schema may probably need some amendment, but what is important is that a skeleton schema has been provided which serves as the foundation for database design. At least the designer knows that nothing has been omitted which is very easily done when the schema is created manually, and the tedium has also been taken out of creating the schemas. Although the system imposes some constraints in this initial release, the schemas can be used to create a database without any amendment, so this is still a very useful tool to the designer.

```
* SCHEMA NAME: SDBC
* DATE CREATED: 04/10/86
* VERSION: 012

SCHEMA NAME IS SDBC

AREA NAME IS  ALL_AREAS
AREA NAME IS  CLIENT_AREA


RECORD NAME IS  AGENT_REC
WITHIN ALL_AREAS

   ITEM NAME IS  AGT_NO                  TYPE IS CHARACTER 8
*  Agent Number


RECORD NAME IS  CLIENT_REC
WITHIN CLIENT_AREA

   ITEM NAME IS  CLIENT_NM               TYPE IS CHARACTER 30
*  Client Name

   ITEM NAME IS  SEX                     TYPE IS CHARACTER 1
*  Sex

   ITEM NAME IS  DOB                     TYPE IS UNSIGNED NUMERIC 8
*  Date of Birth

   ITEM NAME IS  CLIENT_REF              TYPE IS CHARACTER 8
*  Client Reference

   ITEM NAME IS  AG                      TYPE IS UNSIGNED NUMERIC 3
*  Age

   ITEM NAME IS  AGE_ADMT                TYPE IS CHARACTER 1
*  Age Admitted


RECORD NAME IS  CLIENT-STUB_REC
WITHIN ALL_AREAS

   ITEM NAME IS  ADDR                    TYPE IS CHARACTER 30
*  Address

   ITEM NAME IS  POST_CD                 TYPE IS CHARACTER 12
*  Postcode


RECORD NAME IS  FUND_REC
WITHIN ALL_AREAS

   ITEM NAME IS  FUND_CD                 TYPE IS UNSIGNED NUMERIC 3
```

```
*  Fund Code


RECORD NAME IS  PERIODIC_PREMIUM_REC
WITHIN ALL_AREAS

   ITEM NAME IS  AMT_PAY                TYPE IS UNSIGNED NUMERIC 9 -2
*  Payment Amount

   ITEM NAME IS  DT_PREM_LST_COLL       TYPE IS UNSIGNED NUMERIC 8
*  Date Last Collected


RECORD NAME IS  POLICY_REC
WITHIN ALL_AREAS

   ITEM NAME IS  POL_NO                 TYPE IS CHARACTER 9
*  Policy Number

   ITEM NAME IS  POL_STRT_DT            TYPE IS UNSIGNED NUMERIC 8
*  Policy Start Date

   ITEM NAME IS  POL_END_DATE           TYPE IS UNSIGNED NUMERIC 8
*  Policy End Date

   ITEM NAME IS  PROD_CD                TYPE IS CHARACTER 8
*  Product Code

   ITEM NAME IS  PAY_FREQ               TYPE IS UNSIGNED NUMERIC 2
*  Payment Frequency

   ITEM NAME IS  COMP_CD                TYPE IS UNSIGNED NUMERIC 1
*  Company Code

   ITEM NAME IS  ASS_IND                TYPE IS CHARACTER 1
*  Assignment Indicator

   ITEM NAME IS  MAND_COLL_DY           TYPE IS UNSIGNED NUMERIC 2
*  Mandate Collection Day

   ITEM NAME IS  PREM_PAY               TYPE IS UNSIGNED NUMERIC 7 -2
*  Premium Payment


RECORD NAME IS  PRICE_REC
WITHIN ALL_AREAS

   ITEM NAME IS  VALUE_DT               TYPE IS UNSIGNED NUMERIC 8
*  Valuation Date

   ITEM NAME IS  BID_PRICE              TYPE IS UNSIGNED NUMERIC 9 -4
*  Bid Price
```

```
   ITEM NAME IS  OFF_PRICE              TYPE IS UNSIGNED NUMERIC 9 -4
*  Offer Price


RECORD NAME IS  UNIT_REC
WITHIN ALL_AREAS

   ITEM NAME IS  NO_UNITS               TYPE IS UNSIGNED NUMERIC 11 -4
*  Unit Holding

   ITEM NAME IS  VALU_UNITS             TYPE IS UNSIGNED NUMERIC 11 -4
*  Unit Value

RECORD NAME IS CLIENT_POLICY_REC
WITHIN CLIENT_AREA

   ITEM NAME IS  CLIENT_NM              TYPE IS CHARACTER 30
*  Client Name

   ITEM NAME IS  POL_NO                 TYPE IS CHARACTER 9
*  Policy Number

RECORD NAME IS PERIODIC_PREMIUM_FUND_REC
WITHIN ALL_AREAS

   ITEM NAME IS  FUND_CD                TYPE IS UNSIGNED NUMERIC 3
*  Fund Code

   ITEM NAME IS  DT_PREM_LST_COLL       TYPE IS UNSIGNED NUMERIC 8
*  Date Last Collected

SET NAME IS  ALL_AGENT_SET
OWNER IS   SYSTEM
MEMBER IS AGENT_REC
INSERTION IS  AUTOMATIC
RETENTION IS  FIXED


SET NAME IS  ALL_CLIENT_SET
OWNER IS   SYSTEM
MEMBER IS CLIENT_REC
INSERTION IS  AUTOMATIC
RETENTION IS  FIXED


SET NAME IS ALL_OTH_CLIENT_SET
OWNER IS   SYSTEM
MEMBER IS CLIENT_REC
INSERTION IS  AUTOMATIC
RETENTION IS  FIXED
ORDER IS SORTED BY ASCENDING
                              CLIENT_REF
DUPLICATES ARE NOT ALLOWED
```

```
SET NAME IS  ALL_FUND_SET
OWNER IS   SYSTEM
MEMBER IS FUND_REC
INSERTION IS  AUTOMATIC
RETENTION IS  FIXED


SET NAME IS  ALL_POLICY_SET
OWNER IS   SYSTEM
MEMBER IS POLICY_REC
INSERTION IS  AUTOMATIC
RETENTION IS  FIXED


SET NAME IS AGENT_POLICY_SET
OWNER IS AGENT_REC
MEMBER IS POLICY_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER SORTED BY ASCENDING
                                POL_NO
DUPLICATES ARE NOT ALLOWED

SET NAME IS CLIENT_POLICY_SET
OWNER IS CLIENT_REC
MEMBER IS CLIENT_POLICY_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER IS LAST

SET NAME IS POLICY_CLIENT_SET
OWNER IS POLICY_REC
MEMBER IS CLIENT_POLICY_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER IS LAST

SET NAME IS CLIENT_CLIENT-STUB_SET
OWNER IS CLIENT_REC
MEMBER IS CLIENT-STUB_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER IS LAST

SET NAME IS FUND_UNIT_SET
OWNER IS FUND_REC
MEMBER IS UNIT_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER IS LAST

SET NAME IS PERIODIC_PREMIUM_FUND_SET
```

```
OWNER IS PERIODIC_PREMIUM_REC
MEMBER IS PERIODIC_PREMIUM_FUND_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER IS LAST

SET NAME IS FUND_PERIODIC_PREMIUM_SET
OWNER IS FUND_REC
MEMBER IS PERIODIC_PREMIUM_FUND_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER IS LAST

SET NAME IS UNIT_PRICE_SET
OWNER IS UNIT_REC
MEMBER IS PRICE_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER SORTED BY ASCENDING
                                VALUE_DT
DUPLICATES ARE NOT ALLOWED

SET NAME IS OTH_UNIT_PRICE_SET
OWNER IS UNIT_REC
MEMBER IS PRICE_REC
INSERTION IS AUTOMATIC
RETENTION IS MANDATORY
ORDER IS SORTED BY ASCENDING
                                OFF_PRICE
DUPLICATES ARE NOT ALLOWED
```

Figure 8.11

```
* STORAGE SCHEMA NAME: SDBC_STR
* DATE CREATED: 16/12/86
* VERSION: 002

STORAGE SCHEMA NAME IS SDBC_STR FOR SDBC SCHEMA


RECORD NAME IS  AGENT_REC
PLACEMENT IS  SCATTERED USING          AGT_NO

   ITEM NAME IS  AGT_NO                  TYPE IS CHARACTER 8
*  Agent Number


RECORD NAME IS  CLIENT_REC
PLACEMENT IS  SCATTERED USING          CLIENT_NM

   ITEM NAME IS  AG                      TYPE IS UNSIGNED NUMERIC 3
*  Age

   ITEM NAME IS  AGE_ADMT                TYPE IS CHARACTER 1
*  Age Admitted

   ITEM NAME IS  CLIENT_NM               TYPE IS CHARACTER 30
                                           ALLOCATION IS DYNAMIC
*  Client Name

   ITEM NAME IS  CLIENT_REF              TYPE IS CHARACTER 8
*  Client Reference

   ITEM NAME IS  DOB                     TYPE IS UNSIGNED NUMERIC 8
*  Date of Birth

   ITEM NAME IS  SEX                     TYPE IS CHARACTER 1
*  Sex


RECORD NAME IS  CLIENT-STUB_REC
PLACEMENT IS CLUSTERED VIA
                                CLIENT_CLIENT-STUB_SET

   ITEM NAME IS  ADDR                    TYPE IS CHARACTER 30
                                           ALLOCATION IS DYNAMIC
*  Address

   ITEM NAME IS  POST_CD                 TYPE IS CHARACTER 12
*  Postcode


RECORD NAME IS  FUND_REC
PLACEMENT IS SYSTEM DEFAULT
```

```
   ITEM NAME IS  FUND_CD                    TYPE IS UNSIGNED NUMERIC 3
*  Fund Code


RECORD NAME IS  PERIODIC_PREMIUM_REC
PLACEMENT IS SYSTEM DEFAULT

   ITEM NAME IS  DT_PREM_LST_COLL           TYPE IS UNSIGNED NUMERIC 8
*  Date Last Collected

   ITEM NAME IS  AMT_PAY                    TYPE IS UNSIGNED NUMERIC 9 -2
*  Payment Amount


RECORD NAME IS  POLICY_REC
PLACEMENT IS  SCATTERED USING          POL_NO

   ITEM NAME IS  ASS_IND                    TYPE IS CHARACTER 1
*  Assignment Indicator

   ITEM NAME IS  COMP_CD                    TYPE IS UNSIGNED NUMERIC 1
*  Company Code

   ITEM NAME IS  FST_PREM                   TYPE IS UNSIGNED NUMERIC 9 -2
*  First Premium

   ITEM NAME IS  MAND_COLL_DY               TYPE IS UNSIGNED NUMERIC 2
*  Mandate Collection Day

   ITEM NAME IS  PAY_FREQ                   TYPE IS UNSIGNED NUMERIC 2
*  Payment Frequency

   ITEM NAME IS  POL_END_DATE               TYPE IS UNSIGNED NUMERIC 8
*  Policy End Date

   ITEM NAME IS  POL_NO                     TYPE IS CHARACTER 9
*  Policy Number

   ITEM NAME IS  POL_STRT_DT                TYPE IS UNSIGNED NUMERIC 8
*  Policy Start Date

   ITEM NAME IS  PREM_PAY                   TYPE IS UNSIGNED NUMERIC 7 -2
*  Premium Payment

   ITEM NAME IS  PROD_CD                    TYPE IS CHARACTER 8
*  Product Code


RECORD NAME IS  PRICE_REC
PLACEMENT IS CLUSTERED VIA
                        UNIT_PRICE_SET
```

```
   ITEM NAME IS  BID_PRICE                TYPE IS UNSIGNED NUMERIC 9 -4
*  Bid Price

   ITEM NAME IS  OFF_PRICE                TYPE IS UNSIGNED NUMERIC 9 -4
*  Offer Price

   ITEM NAME IS  VALUE_DT                 TYPE IS UNSIGNED NUMERIC 8
*  Valuation Date


RECORD NAME IS  UNIT_REC
PLACEMENT IS CLUSTERED VIA
                        FUND_UNIT_SET

   ITEM NAME IS  NO_UNITS                 TYPE IS UNSIGNED NUMERIC 11 -4
*  Unit Holding

   ITEM NAME IS  VALU_UNITS               TYPE IS UNSIGNED NUMERIC 11 -4
*  Unit Value

RECORD NAME IS CLIENT_POLICY_REC
PLACEMENT IS CLUSTERED VIA  CLIENT_POLICY_SET

   ITEM NAME IS  CLIENT_NM                TYPE IS CHARACTER 30
                                            ALLOCATION IS DYNAMIC
*  Client Name

   ITEM NAME IS  POL_NO                   TYPE IS CHARACTER 9
*  Policy Number

RECORD NAME IS PERIODIC_PREMIUM_FUND_REC
PLACEMENT IS CLUSTERED VIA  PERIODIC_PREMIUM_FUND_SET

   ITEM NAME IS  FUND_CD                  TYPE IS UNSIGNED NUMERIC 3
*  Fund Code


SET NAME IS  ALL_AGENT_SET
MODE IS CALC
   MEMBER IS AGENT_REC
   KEY IS      AGT_NO


SET NAME IS  ALL_CLIENT_SET
MODE IS CALC
   MEMBER IS CLIENT_REC
   KEY IS      CLIENT_NM


SET NAME IS ALL_OTH_CLIENT_SET
MODE IS INDEX
   NODE SIZE IS  685 BYTES
```

```
SET NAME IS  ALL_POLICY_SET
MODE IS CALC
   MEMBER IS POLICY_REC
   KEY IS      POL_NO


SET NAME IS AGENT_POLICY_SET
MODE IS INDEX
   NODE SIZE IS  207 BYTES

SET NAME IS CLIENT_POLICY_SET
MODE IS CHAIN

SET NAME IS POLICY_CLIENT_SET
MODE IS CHAIN

SET NAME IS CLIENT_CLIENT-STUB_SET
MODE IS CHAIN

SET NAME IS FUND_UNIT_SET
MODE IS CHAIN

SET NAME IS PERIODIC_PREMIUM_FUND_SET
MODE IS CHAIN

SET NAME IS FUND_PERIODIC_PREMIUM_SET
MODE IS CHAIN

SET NAME IS UNIT_PRICE_SET
MODE IS INDEX
   NODE SIZE IS  685 BYTES

SET NAME IS OTH_UNIT_PRICE_SET
MODE IS INDEX
   NODE SIZE IS  685 BYTES
```

Figure 8.12

```
* SUB-SCHEMA NAME: SUBSCH
* DATE CREATED: 21/03/87
* VERSION: 004

SUBSCHEMA SUBSCH FOR SDBC SCHEMA



REALM  ALL_AREAS_REALM                  IS  ALL_AREAS
REALM  CLIENT_REALM                     IS  CLIENT_AREA



RECORD NAME IS  AGENT_REC
   ITEM NAME IS  AGT_NO                TYPE IS CHARACTER 8
*  Agent Number



RECORD NAME IS  CLIENT_REC

  GROUP NAME IS AGE DETAILS
   ITEM NAME IS  AG                     TYPE IS UNSIGNED NUMERIC 3
*  Age

   ITEM NAME IS  AGE_ADMT               TYPE IS CHARACTER 1
*  Age Admitted
  ENDGROUP  AGE DETAILS


   ITEM NAME IS  CLIENT_NM              TYPE IS CHARACTER 30
*  Client Name

   ITEM NAME IS  CLIENT_REF             TYPE IS CHARACTER 8
*  Client Reference

   ITEM NAME IS  DOB                    TYPE IS UNSIGNED NUMERIC 8
*  Date of Birth

   ITEM NAME IS  SEX                    TYPE IS CHARACTER 1
*  Sex



RECORD NAME IS  CLIENT-STUB_REC

   ITEM NAME IS  ADDR                   TYPE IS CHARACTER 30
*  Address

   ITEM NAME IS  POST_CD                TYPE IS CHARACTER 12
*  Postcode
```

```
RECORD NAME IS  FUND_REC

   ITEM NAME IS  FUND_CD                TYPE IS UNSIGNED NUMERIC 3
*  Fund Code


RECORD NAME IS  PERIODIC_PREMIUM_REC

   ITEM NAME IS  DT_PREM_LST_COLL       TYPE IS UNSIGNED NUMERIC 8
*  Date Last Collected

   ITEM NAME IS  AMT_PAY                TYPE IS UNSIGNED NUMERIC 9 -2
*  Payment Amount


RECORD NAME IS  POLICY_REC

   ITEM NAME IS  ASS_IND                TYPE IS CHARACTER 1
*  Assignment Indicator

   ITEM NAME IS  COMP_CD                TYPE IS UNSIGNED NUMERIC 1
*  Company Code

   ITEM NAME IS  FST_PREM               TYPE IS UNSIGNED NUMERIC 9 -2
*  First Premium

   ITEM NAME IS  MAND_COLL_DY           TYPE IS UNSIGNED NUMERIC 2
*  Mandate Collection Day

   ITEM NAME IS  PAY_FREQ               TYPE IS UNSIGNED NUMERIC 2
*  Payment Frequency

   ITEM NAME IS  POL_END_DATE           TYPE IS UNSIGNED NUMERIC 8
*  Policy End Date

   ITEM NAME IS  POL_NO                 TYPE IS CHARACTER 9
*  Policy Number

   ITEM NAME IS  POL_STRT_DT            TYPE IS UNSIGNED NUMERIC 8
*  Policy Start Date

   ITEM NAME IS  PREM_PAY               TYPE IS UNSIGNED NUMERIC 7 -2
*  Premium Payment

   ITEM NAME IS  PROD_CD                TYPE IS CHARACTER 8
*  Product Code
```

```
RECORD NAME IS  PRICE_REC

  GROUP NAME IS TOTAL_PRICES
  GROUP NAME IS PRICES
   ITEM NAME IS  BID_PRICE              TYPE IS UNSIGNED NUMERIC 9 -4
*  Bid Price

   ITEM NAME IS  OFF_PRICE              TYPE IS UNSIGNED NUMERIC 9 -4
*  Offer Price
  ENDGROUP  PRICES
  ENDGROUP  TOTAL_PRICES


   ITEM NAME IS  VALUE_DT               TYPE IS UNSIGNED NUMERIC 8
*  Valuation Date



RECORD NAME IS  UNIT_REC

   ITEM NAME IS  NO_UNITS               TYPE IS UNSIGNED NUMERIC 11 -4
*  Unit Holding

   ITEM NAME IS  VALU_UNITS             TYPE IS UNSIGNED NUMERIC 11 -4
*  Unit Value


RECORD NAME IS CLIENT_POLICY_REC

   ITEM NAME IS  CLIENT_NM              TYPE IS CHARACTER 30
*  Client Name

   ITEM NAME IS  POL_NO                 TYPE IS CHARACTER 9
*  Policy Number


RECORD NAME IS PERIODIC_PREMIUM_FUND_REC

   ITEM NAME IS  FUND_CD                TYPE IS UNSIGNED NUMERIC 3
*  Fund Code

   ITEM NAME IS  DT_PREM_LST_COLL       TYPE IS UNSIGNED NUMERIC 8
*  Date Last Collected

SET NAME IS  ALL_AGENT_SET


SET NAME IS  ALL_CLIENT_SET


SET NAME IS ALL_OTH_CLIENT_SET
```

```
SET NAME IS  ALL_POLICY_SET

SET NAME IS AGENT_POLICY_SET

SET NAME IS CLIENT_POLICY_SET

SET NAME IS POLICY_CLIENT_SET

SET NAME IS CLIENT_CLIENT-STUB_SET

SET NAME IS FUND_UNIT_SET

SET NAME IS PERIODIC_PREMIUM_FUND_SET

SET NAME IS FUND_PERIODIC_PREMIUM_SET

SET NAME IS UNIT_PRICE_SET

SET NAME IS OTH_UNIT_PRICE_SET

** END OF SUB-SCHEMA **
```

Figure 8.13

```
* SUB-SCHEMA NAME: CLIENT_ENQ
* DATE CREATED: 31/03/87
* VERSION: 005

SUBSCHEMA CLIENT_ENQ FOR SDBC SCHEMA


*  Subschema is for the following transactions:-
*  Client Enquiry

REALM  CLIENT_REALM    IS    CLIENT_AREA



RECORD NAME IS  CLIENT_REC

  GROUP NAME IS AGE DETAILS
   ITEM NAME IS  AG                                TYPE IS UNSIGNED NUMERIC
*  Age

   ITEM NAME IS  AGE_ADMT                          TYPE IS CHARACTER 1
*  Age Admitted
  ENDGROUP  AGE DETAILS


   ITEM NAME IS  CLIENT_NM                         TYPE IS CHARACTER 30
*  Client Name

   ITEM NAME IS  CLIENT_REF                        TYPE IS CHARACTER 8
*  Client Reference

   ITEM NAME IS  DOB                               TYPE IS UNSIGNED NUMERIC
*  Date of Birth

   ITEM NAME IS  SEX                               TYPE IS CHARACTER 1
*  Sex


SET NAME IS  ALL_CLIENT_SET


SET NAME IS ALL_OTH_CLIENT_SET

** END OF SUB-SCHEMA **
```

Figure 8.14

CHAPTER 9

CONCLUSION

The objective of this thesis was to show that it was possible to develop a computer system that could be used from Data Analysis through to Database Design, the final product being a set of schemas that would be used to actually create the database, a requirement that as one can see has been shown to be feasible. Why is a system like this needed? Well, the initial investigation into the software currently available showed that there was not one tool that provided all of the following facilities.

1. Entity Modelling
2. Detailed Attribute Definition
3. Transaction Definition
4. Normalisation
5. Schema Generation

These tasks are the ones that have to be performed by the Analyst and the Designer when creating a database. This entire procedure is a very long and tedious fact finding task. The time

taken to perform these tasks properly, can vary from several days to a couple of months for a typical complex commercial database. Therefore any automation of the tasks is especially welcome. One must also remember that the database is a very important part of any computer system, therefore it must be designed very carefully. If any errors in the design occur, then it may be impossible to rectify them later if many of the applications require amendment or a re-structure of the database is required. Consequently, it is very important that the design is correct first time. By using an automated system such as this, the analyst and designer are free to concentrate on gathering the information and formulating the design. While some clerical person performs the huge data entry task. Another advantage of using an automated system is the generation of documentation afterwards. Very often when systems are designed manually, documentation is left until the end when the design has stabilized. But often the designer is given another project and the documentation is never completed.

When the market was originally investigated some three years ago, the only products that were available were Data Dictionary systems. These were very comprehensive systems that provided the ability to define detailed attribute definitions and possibly transaction definitions. Not surprisingly all of these tools ran on either mini or mainframe computers.

It was then interesting to see some two years later the introduction of a number of tools that provided the Entity Modelling capability. The trend now seemed to be towards creating systems

that ran on Personal Computers like the IBM PC, rather than large computers. Fortunately, the work being done here was still valid because these new systems were only addressing the problems of Entity Modelling. Admittedly the facilities provided were very impressive, since these systems were using the small personal computers, which have very good graphics capabilities.

What had now resulted was a situation where the design team would use a personal computer for their Entity Modelling, a mini/mainframe computer for their Data Dictionary, and then manually verify the transactions and create the schemas by hand. There still was not one fully integrated system which performed all of these tasks, which is just what this system was trying to achieve.

Why was there still not a fully automated tool available? This is probably due to the limited market for a product such as this and the number of man years that would be required to complete the system. In the course of my job in the computer industry, I have visited a number of database installations. Of all the sites that I have visited, only one had an entity modelling tool and none had a data dictionary system. References will show that the number of data dictionary licenses issued worldwide is small compared with the number of databases installed. [INTDIR84]

It is also likely that the suppliers of computer software may think that the vast majority of users who might design a CODASYL database have done so already. At the moment interest seems to be concentrated on the relational, rather than the network model. This

is a rapidly expanding market and many companies who have traditionally large network databases, are now considering smaller relational models for some of their systems. Since a true relational database is defined according to Codd's rules, it should be easier to develop a schema for a relational database. In fact DATADESIGNER has recently been modified to perform just this function, and recently the 1986 British Computer Society Database Conference concentrated entirely on relational database technology.

One of the advantages that sales people try to emphasize with a relational database is that a Database Administrator is not required to design and maintain the database. With all this effort being concentrated on relational technology, some people are beginning to think that the days of the CODASYL database are over, despite the fact that most CODASYL databases will generally out-perform a relational database. What the thesis is trying to emphasize is that with this tool, CODASYL databases can be developed just as easily, therefore an expensive Database Adminstrator does not have to devote 100% of his time to looking after the database, and non-experienced staff can be used for the data entry, leaving the Database Administrator time free to concentrate on the more important tasks.

A fully automated database design system has been created, but unfortunately due to the size of the system, certain features have had to be omitted so that it could be completed in a reasonable time scale. For example the graphics capability of this system is limited because no graphics software was available. There are not

all the features that one would normally see in a Data Dictionary, otherwise the entire thesis could have concentrated on developing a Data Dictionary system. Despite this, the system is still completely functional.

The starting point for this system is the Entity Modelling stage, and the facilities available compare very favourably with the software that is now available. The system provides a very easy method of defining the model, which some users may prefer to actually drawing. Where the system needs to be enhanced is in the presentation of the graphical display and hardcopy output of the model. The ability to show the role of either the owner or member entity would be desirable, instead of just the owner which it does at present. One feature of this system is the ability to define both the high level and the low level entity diagrams. Currently all of this detail is displayed on the model, although there is no reason why the system should not be enhanced to display only certain parts of the model. Nevertheless, when the company sponsoring the research used this part of the system for the first time, they found data entry very straightforward, quick and easy. The entire entity model comprising of some sixty entities was defined very quickly. Subsequently it has been modified and this part of the system is periodically used as changes to the model occur. This illustrates one important feature of this system, the ability of the various parts of the system to be used independently.

Once the entities have been defined, our model is not complete until all of the attributes of the entities have been defined. A rather unusual approach was taken by this system to capture the data. Instead of asking the user to define the attributes, they were asked to enter all the inputs and outputs of the system, grouping them all together under the general heading of 'documents'. Then as each document is defined, the attributes are automatically allocated to an entity if at all possible. The technique used is a simple one, which works very well, but none of the software tools seemed to use this technique for capturing the data.

One possible drawback to this system is that data entry for an attribute cannot be completed as one task. It is a two step task comprising of initial data entry and the full data entry. The initial data entry step was welcomed because often one does not want to specify all the detail until one is sure about what is to be included in the system. Therefore providing only skeleton information quickens the design process. Then using the Document Action Report to identify problems in the model, allows the system to decide what is incorrect rather than trusting the analysts eye. So what has been provided here is a tool that tries to help rather than just document.

If one could improve upon this part of the system then the ability to store a physical copy of the document so that it could be reproduced would be worth considering. At the moment the analyst knows the attributes that are on the document, but has no idea of

their respective positions. This facility would be particularly useful in the area of screen definitions. It is also hoped to improve upon the rather inflexible approach that is used to define descriptions. Nevertheless, this whole area of Entity Modelling and Document Definition provides the design team with a set of very comprehensive tools, which despite its weaknesses performs very well indeed.

One useful by-product that has resulted from this part of the system is a Data Dictionary, but please remember that Commercial Data Dictionary systems have been developed over many man years, and this system was never designed to be a Data Dictionary, therefore in the limited time available, it was impossible to include all of the features that one would expect to see, neither was it felt necessary. But, what has been provided for in this system, is the ability to define comprehensive information about an attribute, although in this release there is no facility to hold details about more than one version of an attribute.

Some of the useful featuers of this system include the ability to enquire upon the data using either the whole or part of the computer or attribute name, and define the validation procedures for an attribute. This information can then either be kept as text or included directly in the schema, provided it is in the correct format. In the future it is hoped to be able to remove the demand for this rigid syntax on the validation procedures.

Once Entity Modelling is complete, the model must now be validated. Typically this is a process done by hand, which is achieved by taking each transaction that the model must process and passing it through the model. Now although a few of the systems allowed the transactions to be described, beyond this point no other facilities were available. Consequently this provided a great opportunity to show that it is feasible to describe the transaction in any format, and then using a simple natural language processor, validate the description against the model that has been previously defined. The natural language processor is a very simple one because in this application the detail that will be described is limited to information which is already in the system.

It uses all of the words in the system as its dictionary, but additional words may be input if required. Therefore all the information that has been previously supplied such as entities and attributes becomes the dictionary. As the purpose of the transactions is to describe the flow through the model. Provided it is described correctly, all the information should already be in the system. If it does come across a word that it cannot find then it will mean that either the description is wrong or something is missing from the model. Consequently it is not necessary to build a very complex natural language processor.

The natural language processor performs two tasks, the first is to validate against all the information that has been entered previously, so that hopefully anything that has been overlooked will be highlighted on the error report. Secondly the flow of the

transaction is displayed for the analyst to check that the correct details have been entered. In fact by generating the flow, the next step would be to actually generate the code required for the description of the transaction that has been supplied. The text input to this system could easily be used as input to some fourth generation language program generator, such as that described by Luker and Brown, which would produce the code necessary to perform the transaction. [LUKERB]

This whole area of transaction verification has not been easy and an entire thesis could have been developed on solving just this problem. The objective was to prove that what was being asked was feasible, and this has certainly been show. The system works very well, it is unfortunate that the time taken to verify a transaction can take many seconds. Consequently this task had to be withdrawn from the online facilities so that the users were not frustrated with the response times. It is probably feasible that the algorithms used by the natural language processor could be improved to improve system performance. Given time this entire area could be enhanced to analyse the description further. However, this initial release performs well and is more than adequate for verifying the model.

Normalisation was included because it does have it place in database design. Unfortunately it is one of those tasks that works very well in theory, but in practice can often cause terrible performance problems. Therefore from my own personal experience of designing Commercial databases, it was decided to let the designer

choose whether normalisation is suitable for their system. Since normalisation is particularly relevant to databases based on the relational model, and this system is for databases based on the network model, no functionality is lost in the database. Because normalisation is not such a critical task only normalisation to first normal form is performed.

Finally the system creates the schemas necessary to build the database, and it has been found that the schemas generated only required minor amendment. Multiple versions of the schemas may be retained so it is possible to see the changes that have been applied. The only restrictions with the schemas is that it is not possible to specify the order in which the attributes will appear on the schema. If a set is sorted the rules for duplicates cannot be specified and there is no ability to define a check clause between records and sets. However, these problems are minor compared to the work that the system has done. These schemas were never envisaged as being the master database definitions that would never require amendment. Instead their purpose is to ensure that nothing important has been omitted and they are there for the designer to use his skills to amend to suit the needs of the application. As was stated at the outset, this software does not set out to replace the database designer, it is merely a tool used to assist during database design.

One of the important features of this system is the consistency checks that are continually being performed upon the data eg. an attribute cannot be assigned to an entity until the

entity has been defined. Attributes or entities referenced in a transaction must have been previously defined or the transaction will not be considered to be verified. This then means that inexperienced database design staff can enter data into the system and the Database Administrator knows that it will not be accepted unless it is correct. In fact the company sponsoring the research used one of the secretaries for the initial data entry, which proved to be very successful. She was not very familiar with computer systems, but she found the system very easy to use. This meant that the Database Administrator team was left to concentrate on changes to the database, while the secretary set up the Data Dictionary. Hence this system was brought into use considerably quicker.

The primary objective was to create a prototype online user-friendly system that creates database schemas, and this has certainly been achieved. All of the facilities provide a good response and sufficient information is available on each screen to prevent excessive queries on the database. Although this system has not been used in its entirety to develop a database in a commercial environment. Its modularity means that the system does not remain dormant, because parts of it can be used in isolation. This is happening in the company that sponsored the research, currently they are actively using all of the features up to transaction definition. It is hoped that one day this system will be used to actually develop a real commercial database.

APPENDIX A

REFERENCES

[ALBANO] Albano Antonio et al DIALOGO:An Interactive Environment for Conceptual design in Galileo. Methodology & Tools for Database Design 1983

[ALTZENI] Altzeni P et all. INCOD-DTE; A system for interactive conceptual design of data, transactions and events. Methodology & Tools for Database Design 1983.

[BACHMAN] Bachman C.W. 'The Role Concept in Data Models' 3rd Int Conf on Very Large Databases Tokyo 1977

[BACH C] Bachman C.W. "The Role Data Model Approach to Data Structure International Conferences on Databases 1980

[BCS82] Data Analysis Update BCS DATABASE 82 Edited by Geoffrey J. Baker

[BCS83] Data Dictionary Update BCS DATABASE 83 Edited by Geoffrey J. Baker

[BCSWRP] British Computer Society Data Dictionary Working Party Report. March 1977

[BCS85] Data Analysis in Practise BCS DATABASE 85 Edited by Simon Holloway

[BRACCHI] Bracchi G et al. A set of Integrated Tools for the Conceptual Design of Database systems. p 181 - 204 Methodology & Tools of Database Design 1983

[CDDDEC] VAX Common Data Dictionary Users Guide AA-AY81A-TE

[CHEN] Chen P.P.S. 'The Entity-Relationship Model : Towards a Unified View of Data' ACM Transactions on Database Systems Vol 1 1976 p9-66

[CHEN 80] Chen P.P.S 'The Entity-Relationship Approach to Systems Analysis and Design. Proceedings of the International Conference on Entity-Relationship Approach to Systems Analysis and Design. Los Angeles 1979

[CODASYL] CODASYL Data Description Language.
Journal of Development 1981.
Canadian Government Publishing Centre

[CODD] Codd E.F. 'Extending the Database Relational Model'
Proceedings of ACM-SIGMOD 1979 Boston

[COLOMB] Colombetti M et al. A Natural Language Reasoning System for the Analysis of Data Requirements. p 149-179
Methodology & Tools for Database Design 1983

[COMPDD] Data Dictionaries Computing Feb 7 1985

[DATECJ] An Introduction to Database Systems C. J. Date

[DATIBM] Data Analysis. IBM United Kingdom 1981

[DBMS] Database Management Systems. Practical Aspects and Their Use Edited by R.A. Frost

[DELTA] Delta Software Tools

[DATADES] Data Designer Introductory Guide.
Data Design Incorporated. 1982

[DSGNMGR] DESIGNMANAGER Database Design & Information Modelling Release 2.7

[ELLIS] Twenty Years of Data Analysis. Harry Ellis
Data Analysis in Practise BCS DATABASE 85

[FLORY] Flory, A. An Approach to Designing an Entity - Relationship Schema. Entity Relationship Approach to Systems Analysis and Design p143-144

[FROST] Database Management Systems - Practical Aspects of Their Use Edited by R.A. Frost

[GARDE] Serviced et Ingenierie et Informatique General Presentation

[HALL] Hall, John. The LBMS Systems Development Method
BCS Data Analysis Update p 41-69

[HARRIS] Harris, Mary Dee Introduction to Natural Language Processing 1983

[HOLLOW] How to Manage the Information Resource.
Proceedings from the Conference for the BCS Association

for the Blind. Leeds 1985. Edited by Simon Holloway

[HUBBARD] Hubbard, George U Computer Assisted Database Design

[IBM] Data Analysis Manual IBM UK 26-8101

[ICLDDS] ICL Data Dictionary System for VME 2900 (DDS 600)

[IEW] Information Engineering Workbench 1985
KnowledgeWare Inc.

[INTDIR81] International Directory of Software 1981. Hoskyns

[INTDIR84] International Directory of Software 1984. Hoskyns

[INTEL] Intellect Software International. Wembley UK

[ISM] Information System Methodologies

[JUKES] Jukes, Dick Data Analysis - Practical and Technical
Problems. BCS Data Analysis Update p 1 - 39

[KENT] Kent W. The semantics of records.
Fundamentals of Data Analysis,Design and Documentation.
BCS DATABASE 85

[LECHNER] Lechnert, Wendy G and Ringle, Martin H
Strategies for Natural Language Processing 1982

[LOMAX] Lomax. J.D. NCC Data Dictionary Systems

[LUKERB] Luker. P.A. and Burns. A.
Program Generators and Generation Software
Computer Journal Aug 1986 p315-321

[MAYNE] Mayne Alan. Data Dictionary Systems - A Technical Review
NCC Publications 1984

[MARTIN] Computer Data-Base Organization. James Martin

[MILLER] Miller David R. Strategic Data Analysis.
Data Analysis in Practise BCS DATABASE 85

[MTDBO] Methodology & Tools for Database Design . Editor S. Ceri 198

[OLLE TW] The CODASYL Approach to Database Management T.William Olle

[PARKIN] Parkin A. Data Analysis and System Design by
Entity-Relationship Modelling. Computer Journal
Vol 25 No 4 1982 p 401-409

[SCHWEB] Abstraction Capabilities and Invariant Properties Modelling
within the Entity-Relationship Approach

Entity Relationship Approach to Systems Analysis and Design

[TOZER] Tozer, Edwin E. A review of Current Data Analysis Techniques
Data Analysis in Practise BCS DATABASE 85

[TSILOC] Tsichritiz, Dionysios C & Lochovsky, Frederick H
Data Models 1982 Prentice Hall

[WALTZ] David L. The State of the Art in Natural Language Understand
Strategies for Natural Language Processing p 3-32 1982

[WHITE] White David I. Effective Data Analysis
BCS Data Analysis Update p 151 - 153

[WIDGE] Widger John & Russell-Clark, Tony Data Analysis
BCS Data Analysis Update p 71-85

[VAX] Trademark of the Digital Equipment Corporation