A SYSTEM FOR AUTOMATIC GENERATION OF RELATIONAL DATA BASES

BY

MEIR COHEN

B.S., TEL AVIV UNIVERSITY, ISRAEL, 1979

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1981

Approved by:

Major Professor

TABLE OF CONTENTS

Introduction

Data base management system (DBMS) technology has achieved acceptance and wide application. This is demonstrated by numerous surveys and the availability of several textbooks dealing exclusively with the topic. However the usefulness of a DBMS is directly related to the way in which the actual data base was designed. Numerous papers have been published dealing with the design phase of data base implementation in an organization. Some agreement exists among these researchers as to the categorization of the design process in two of these phases- the logical design and the physical design. However, agreement has yet to be reached about the overall design process.

Most researchers agree that design should be automated as much as possible. One of the basic components in the design is a descriptive mechanism that represents the information input and the results of each design step. Hollist (Hollist [80]) called this mechanism the Design DBMS.

Among the characteristics required in a Design DBMS, one can identify the ability to detail data names, types, domains and ranges. In addition the design DBMS should be easy to operate and the processes which it automated as possible. The commands should be flexible and in an English-like form.

This report deals with the implementation of the Document Handler, a tool in the logical design of data bases on the

Interdata 8/32 at Kansas State University. The methodology described in the second section is based mainly on the work of Fisher (Fisher [79]). The third section gives a detailed description of the structure and operation of the Document Handler. An extensive example is given in the fourth section. This demonstrates the capabilities of the Document Handler.

Data Base Design - The Methodology

Introduction

One can divide the design process into three general processes:

1) Organization survey and general design
2) Detailed design
3) Key Specification and Schema Translation

Although there seems to be a clear cut distinction between these processes, this is not the case. It will often be required to go back from process 2 to process 1 and from process 3 to process 2. In all these processes, the design DBMS is of great help.

The following sections will describe the design in more detail.

Organization Survey and General Design

The first thing designers ought to do is understand the structure of the organization, its operation, its problems and its aims. This can be accomplished by a series of lectures and interviews with the management of the organization under investigation.

When this understanding is reached, the designers can collect all the documents currently used by the organization. The documents and all the columns in them are entered into the design DBMS with a specification of their use as input, output or resident documents. Attributes such as type, range of values and security constraints can be associated to the columns.

At the conclusion of this step, the design DBMS will summarize and format the information so that it can be used in the next step in the design process.

Detailed Design

Three activities take place in the detailed design :

1) Deletion of synonym column names

2) Removal of insignificant columns

3) Solving undeclared output columns

These activities are not necessarily done in this order. The documents and the columns in them are scanned one by one and a decision is made as to the appropriate action to be taken.

It might be necessary to go over the information time and again in this step to ensure that the design is complete and that the integrity of the data is maintained. It is certainly better to repeat the process at this time and not later, when the cost would be enormous.

Deletion of Synonym Columns

In many cases, different names are used to specify the same entity or attribute of an entity in the organization. This is also reflected in the documents of that organization. For example, the name "DATE" in document A might refer to "SHIPPING DATE", as it appears in document B. It is apparent to the user of document A that this is the meaning of "DATE", these two names should be identical and hence "DATE" should be renamed "SHIPPING DATE".

A one-to-many or many-to-many synonym might also occur. For example, the column "ADDRESS" in document C contains the columns "STREET", "CITY", "STATE" and "ZIP CODE" in document D. In this case, a decision has to be made if the amount of detail in document D is required. According to that decision the columns "STREET", "CITY", "STATE" and "ZIP CODE" will be replaced by "ADDRESS" or vice versa.

Removal of Insignificant Columns

Some columns cannot be included in the data base because of their nature. Other columns may not be significant enough for the organization. Such a column may be a signature on a receipt, which is important in the commercial respect but is insignificant to the organization as far as the data base is concerned. These columns are deleted from the information kept in the design DBMS.

Solving Undeclared Output Columns

A check is done to ensure that all the columns that appear in output documents exist in either input or resident documents. When a column is found to appear only in output documents an investigation is conducted about its origin. In some cases the problem can be solved by finding a synonym. In other cases, the

column may be calculated from another column, thus an artificial document has to be added with all the columns used in calculating the problematic column. The addition of such a document may require the revision of all the information, because a number of new columns may be introduced in this step.

Keys Specification and Schema Translation

The next step is to specify the keys for each document. A candidate key may be considered every combination of columns in which all of the columns in one particular document may be sorted or filed. In some documents, no key exists and a synthetic and unique key must be added.

If more than one candidate key exists, the specfication of all the candidates is done by adding a number of duplicates of the document, each one with a distinct candidate key.

The design of the data base is now complete and the schema translation may begin. This process involves the conversion of the information gathered and manipulated into the particular data model selected by management or the design team. Three major data models currently exist hierarchical, network and relational models. We will deal only with the relational model, as it is the most straightforward one. A more comprehensive discussion may

be found in Hollist [80].

Once the keys are defined, the columns in any document may be treated as functional dependencies and can be used as input to Bernstein's algorithm (Bernstein [76]). This results in a schema in third normal form.

Informaly stated, the steps of Bernstein's algorithm are as follows:

1) Eliminate from the functional dependencies those items that can be derived from other functional dependencies

2) Eliminate from the set of functional dependencies those functional dependencies that can be derived from the set

3) Group the remaining functional dependencies such that all have identical left hand sides

4) Merge the groups that have equivalent left hand sides

5) Remove transitive dependencies from among the data items

6) Construct relations using the groups of functional dependencies

The resulting relations will embody the semantic nature of the data, because of the methodology used to arrive at the functional dependencies.

System Description

System Entities

The system has four related entities :

1) Documents,

2) Columns,

3) Document - Attributes,

4) Column - Attributes.

Each document has a set of columns and a set of document - attributes. A document is uniquely identified by its name.

Each column is owned by one document, although the same column name might appear in several documents. As a result, a column is uniquely identified by its name and by the name of the document in which it occurred. Each column might have a set of column attributes.

An attribute is uniquely identified by being a document - attribute or a column - attribute and by its name. Each attribute has a set of values called the components of the

attribute. The attributes are declared and treated quite the same as user defined scalar types in PASCAL.

Figure 3.1 Illustrates the entities described above and the relations between them.

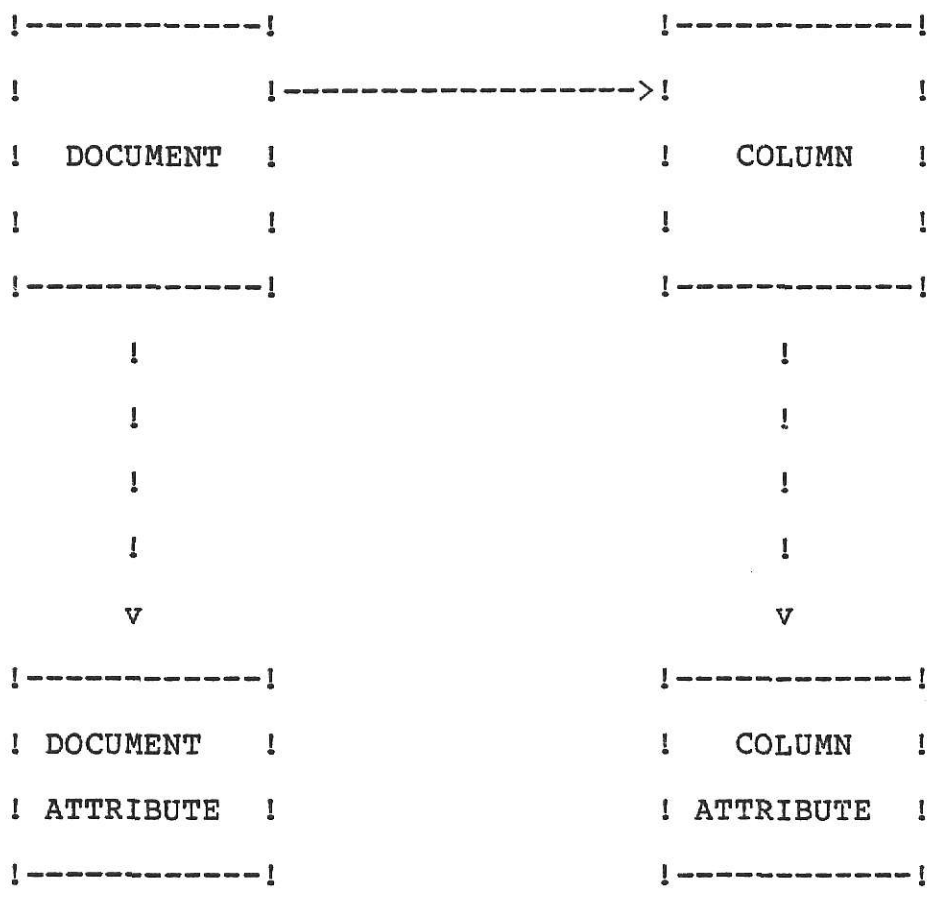


figure 3.1 : The Entities in The System

Files In The System

The system, besides being interactive and thus using a terminal for interactive input and output, deals with three additional files at the same time. These files accumulate data intended for hardcopy printout, for input data to Bernstein's algorithm and for data to be saved so that the status of the system can be restored from session to session. Figure 3.2 illustrates the files in the system and their interaction with the program.

The treating of files and I/O in the system is somewhat difficult and restrictive. The PASCAL software currently implemented on the INTERDATA 8/32 at Kansas State University does not support the standard PASCAL I/O as it is described in the literature (for example JENSEN [73]). A great deal of effort was spent trying to avoid the SVC type of I/O so that the program would keep the portability characteristics provided by the use of PASCAL as a programming language. The retrictions imposed on the system because of the limeted PASCAL I/O will be detailed later when the various commands are discussed.

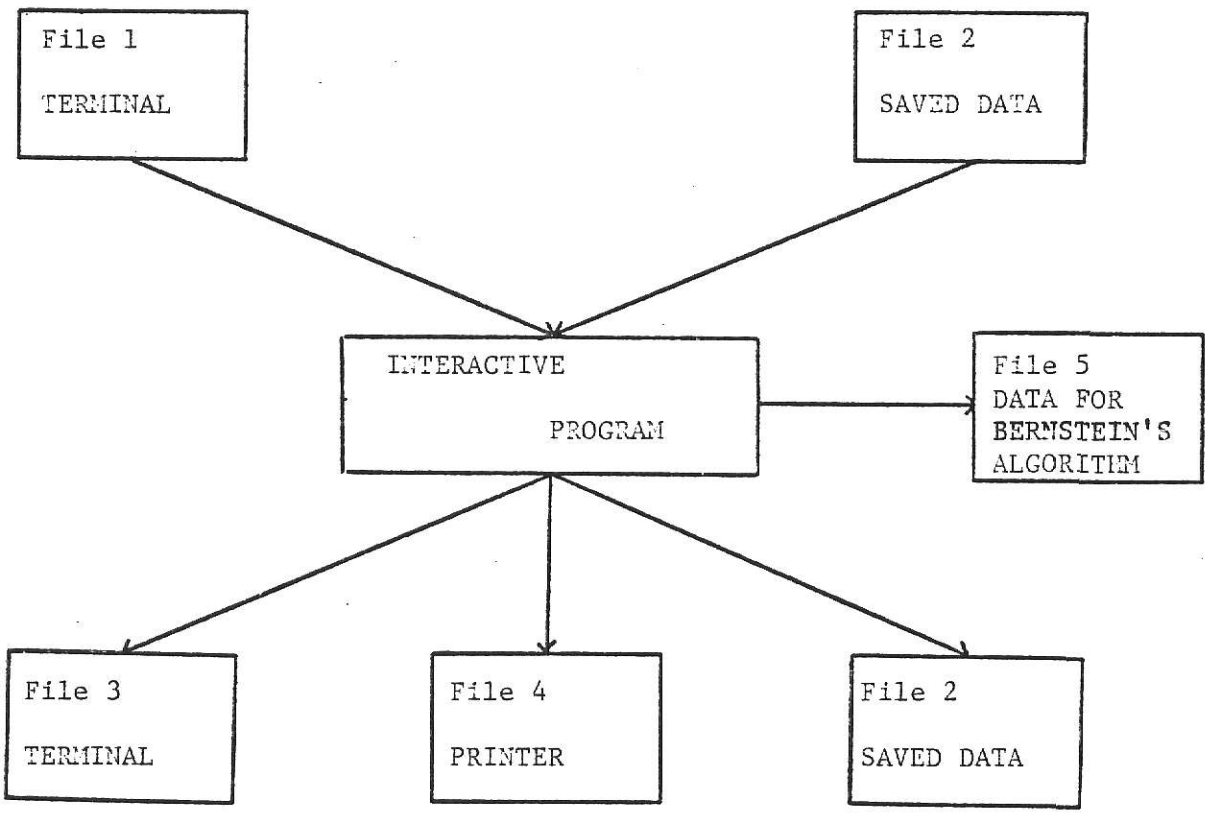


figure 3.2 : Files In The System

Identifiers

Two kinds of identifiers are used in the system:

1) Reserved Identifiers

2) User defined identifiers (names).

Table 3.1 lists the reserved identifiers. Use of any of these identifiers (where a name is expected) will result in a syntax error and the appropriate error message will be displayed. An occurrence of a reserved identifier in a list of names will terminate the scan of the command. Table 3.2 lists the abbreviated form of some of the reserved identifiers.

Names start with a letter and may contain up to 27 more letters, digits or underscores. Any other character truncates the name and might cause a syntax error, depending on the command and on the location of that character. The bound of 28 characters per name may be changed by modifying a program constant.

Identifiers may not be continued over the end of the line, since a carriage return and end of line are treated as blanks.

1) AD	19) FIND	37) RT
2) ADD	20) FROM	38) RUNNAME
3) AL	21) IN	39) RU
4) AS	22) KEY	40) SAVE
5) AT	23) KY	41) SELECT
6) ATLIST	24) LIST	42) SIMILAR
7) ATTRIBUTE	25) LS	43) SL
8) CL	26) N	44) SM
9) COLUMN	27) NO	45) SV
10) DC	28) PE	46) TE
11) DEFINE	29) PR	47) TEACH
12) DELETE	30) PREPARE	48) TO
13) DF	31) PRINT	49) WITH
14) DL	32) REMOVE	50) WITHOUT
15) DOCUMENT	33) RENAME	51) XR
16) EN	34) RETRIEVE	52) XREF
17) END	35) RM	53) Y
18) FI	36) RN	54) YES

Table 3.1 : Reserved Identifiers

1)	ADD	AD	13)	PREPARE	PE
2)	ATLIST	AL	14)	PRINT	PR
3)	ATTRIBUTE	AT	15)	REMOVE	RM
4)	COLUMN	CL	16)	RENAME	RT
5)	DEFINE	DF	17)	RETRIEVE	RT
6)	DELETE	DL	18)	RUNNAME	RU
7)	DOCUMENT	DC	19)	SAVE	SV
8)	END	EN	20)	SELECT	SL
9)	FIND	FI	21)	SIMILAR	SM
10)	KEY	KY	22)	TEACH	TE
11)	LIST	LS	23)	XREF	XR
12)	NO	N	24)	YES	Y

Table 3.2 : Abbreviations for reserved identifiers

Command Structure

Since the system is intended for interactive work to be done most of the time using a portable terminal carried by the data base designer to the location of the investigated organization, a great deal of emphasis was put into the design of easy-to-use commands.

The availabilty of the system to receive the next user's command is indicated by the prompt

COMMAND :

and a bell ring.

A program constant bounds the length of the command to 1600 characters. Experiments have shown that this size is more than satisfactory. If the user happens to go over that bound, an error message is displayed, the command is ignored, and the buffer that accumulates the characters is reset.
Most of the terminals have a backspacing mechanism that deletes characters on the same input line. Since the commands in this system often will be longer than one input line, and it is desirable to be able to delete any number of characters, the system provides two means of doing this:

1) The character @ deletes the last input character (if there is one). If deleting characters from a previous line, that is, after a carriage return is pressed, an extra @ is required to delete the blank caused by the carriage return.

2) The character & deletes all the previously input command and resets the command buffer.

The end of the input command is indicated by a semicolon. A semicolon cannot be used as a separator between command elements.
The identifiers used in the command should be separated by blanks or commas. Whenever one blank appears, any number of blanks may appear. Blanks and commas are interchangeable.

The following pages will describe the various commands in the system.

The Commands

Nineteen Commands are supported by the system. In general, these may be categorized into five categories.

1) Commands that change the data stored in the system :
   DEFINE
   ADD
   RENAME
   DELETE
   REMOVE
   RUNNAME

2) Commands that save and retrieve the data :
   SAVE
   RETRIEVE

3) Commands that list the data according to various criteria :
   DOCUMENT
   COLUMN
   LIST
   XREF
   ATLIST
   SELECT
   FIND

4) Commands that prepare the data for Bernstein's algorithm :
   PREPARE
   KEY

5) Other commands :
   TEACH
   END

The following sections will explain the commands one by one. For the details of the execution of the commands, please refer to the user manual. Appendix A - Commands' syntax graphs might prove helpful after an understanding of the commands is achieved.

DEFINE

DEFINE is used to declare column or document attributes.

Command's structure :

1) DEFINE DOCUMENT [ATTRIBUTE] name (list_of_components)
2) DEFINE COLUMN [ATTRIBUTE] name (list_of_components)

where the list of components is a sequence of names separated by blanks or commas. The word ATTRIBUTE is used to achieve English like form and thus may be removed.

Examples :
1) DEFINE DOCUMENT ATTRIBUTE location (inp,int,out) ;

2) DEFINE COLUMN ATTRIBUTE type (string , integer) ;

Error Messages :

1) In case of syntax error, the message
**** DEFINE - SYNTAX ERROR
will be displayed.

2) If the attribute is already defined, The message
ERROR - ATTRIBUTE ALREADY DEFINED
will be displayed.

ADD

ADD is used to add a document to the system, or a column to an existing document, or an attribute to an existing document or an existing column within a document.

Command's Structure :

1) ADD DOCUMENT name (list_of_attributes)
2) ADD COLUMN name (list_of_attributes) TO [DOCUMENT] name
3) ADD ATTRIBUTE name.name TO DOCUMENT name
4) ADD ATTRIBUTE name.name TO COLUMN name IN [DOCUMENT] name

In forms 1 and 2, the meaning of list_of_attributes is zero (in which case the parentheses may be removed) or more elements of the form name.name, where the first name is the attribute name and the second is a name of a component of that attribute. In the case of form 1, a message will be displayed, after a check has been done to ensure that the new document has a unique name, requesting the names and attributes of the columns for this document. The input from the user should be a list of elements, where each element is of the form name (list_of_attributes). In forms 2,3 and 4, no further data is requested from the user.

Examples :

1) ADD DOCUMENT invoice (location.inp);
Computer response :
INSERT COLUMN NAMES FOR DOCUMENT : location
User's response :
column1 (type.string), column2 (type.integer) ;

2) ADD COLUMN column3 (type.string,type.integer) TO DOCUMENT invoice ;

3) ADD ATTRIBUTE location.out TO DOCUMENT invoice ;

4) ADD ATTRIBUTE type.string TO COLUMN column1 IN DOCUMENT invoice ;

Error Messages :

1) Any syntax error will result in the error message

**** ADD - SYNTAX ERROR

2) If the user tries to add a document that already exists the message
DOCUMENT ALREADY EXISTS
will be displayed.

3) If the user refers to a document that does not exist in the system the message
DOCUMENT name DOES NOT EXIST
will be displayed.

4) If an illegal column name appears in the list of columns, in form 1, the message
WRONG COLUMN IN COLUMN LIST
will be displayed.

5) If the user refers to a column that does not exist in a given document the message
COLUMN name DOES NOT EXIST IN DOCUMENT name
will be displayed.

6) If the user tries to add a column that already exists within a given document, the message
ERROR - COLUMN name ALREADY EXISTS
will be displayed.

7) If an attribute in a list of attributes was not previously defined, the message
ATTRIBUTE name NOT FOUND
will be displayed.

8) If a component of an attribute is not found then the message
COMPONENT name NOT FOUND
will be displayed.

9) If an illegal element appears in the list_of_attributes, then the message
ATTRIBUTE LIST ERROR
will be displayed.

10) If the user tries to add an attribute and the attribute already exists, the message
ATTRIBUTE ALREADY EXISTS
is displayed

DELETE

Delete is used to delete a document with all its columns. It is also used to delete a column within a certain document, or delete an attribute from a document or a column. The structure of DELETE is very similar to that of ADD, with the main difference being the absence of the list_of_attributes.

Command's Structure :

1) DELETE DOCUMENT name
2) DELETE COLUMN name FROM [DOCUMENT] name
3) DELETE ATTRIBUTE name.name FROM DOCUMENT name
4) DELETE ATTRIBUTE name.name FROM COLUMN name IN [DOCUMENT] name

Examples :

1) DELETE DOCUMENT invoice ;
2) DELETE COLUMN column3 FROM DOCUMENT invoice ;
3) DELETE ATTRIBUTE location.out FROM DOCUMENT invoice ;
4) DELETE ATTRIBUTE type.string FROM COLUMN column2 IN DOCUMENT invoice ;

Error Messages :

1) In case of any syntax error, the message
**** DELETE - SYNTAX ERROR
will be displayed.

2) If the user refers to a document and the document does not exist, the message
DOCUMENT name DOES NOT EXIST
will be displayed

3) If the user refers to a column within a certain document and the column is not found, the message
COLUMN name DOES NOT EXIST IN DOCUMENT name
will be displayed.

4) If the user refers to an attribute that does not exist, the message
ATTRIB/COMPON NOT FOUND
will be displayed.

RENAME

RENAME provides the facility to change the name of a document or a column within a document.

Command's Structure :

1) RENAME DOCUMENT name AS name
2) RENAME COLUMN name AS name IN [DOCUMENT] name

Examples :

1) RENAME DOCUMENT invoice AS customer_invoice ;
2) RENAME COLUMN columnl AS DATE IN DOCUMENT invoice ;

Error Messages :

1) In case of syntax error the message

**** RENAME - SYNTAX ERROR
will be displayed.

2) If the user refers to a document that does not exist, the message
DOCUMENT name DOES NOT EXIST
will be displayed.

3) If the user refers to a column that does not exist in a certain document, the message
COLUMN name DOES NOT EXIST IN DOCUMENT name
will be displayed.

REMOVE

REMOVE deletes an attribute from either the document's or column's attribute lists, and deletes all the occurrences of that attribute.

Command's Structure :

1) REMOVE DOCUMENT [ATTRIBUTE] name
2) REMOVE COLUMN [ATTRIBUTE] name

Examples :

1) REMOVE DOCUMENT ATTRIBUTE location ;
2) REMOVE COLUMN ATTRIBUTE type ;

Error Messages :

1) In case of a syntax error, the message
**** REMOVE - SYNTAX ERROR
will be displayed.

2) If the attribute was not found, then the message
ERROR - ATTRIBUTE NOT FOUND
will be displayed.

RUNNAME

RUNNAME displays and modifies the name of the run. This name identifies the output as per investigated system and specific run.

Command's Structure :

1) RUNNAME
2) RUNNAME name

The first form displays the current runname (initially there is no name). The second form changes the current name (if any)

to the new name.

Example :

```
RUNNAME payrol_sub_system
```

Error Messages :

If an illegal name is given, the message
**** RUNNAME - SYNTAX ERROR
will be displayed.

SAVE

SAVE saves the data for further sessions. Save may be executed any number of times during the session. The saved data is written using logical unit 2. All the file operations are performed automatically.

Command's Structure :

SAVE

Error Messages :

No error Messages.

RETRIEVE

RETRIEVE may be executed once in a session. The execution of RETRIEVE causes the addition of all the data that existed when the last SAVE was executed to the existing data. The data is read from the saved data file using logical unit 1. All the file operations are done automatically by the program.

Command's Structure : RETRIEVE

Error Messages :

The only error messages that may be displayed while RETRIEVE is being executed are these of ADD, DEFINE, KEY and RUNNAME, that result from errors in the data file.

DOCUMENT

DOCUMENT is used to display the names of the documents that exist in the system, ordered according to their first letter. If preceded by PRINT, the output will be diverted to the printer.

Command's Structure :

DOCUMENT

Error Messages :

No error messages.

COLUMN

COLUMN is used to display the names of all the columns that exist in the system, ordered according to their first letter. If preceded by a PRINT, the output will be diverted to the printer.

Command's Structure :

COLUMN

Error Messages :

No error messages.

LIST

LIST is used to display one or more documents with all the columns and all the attributes that relate to the documents and columns listed. If preceded by a PRINT, the output will be diverted to the printer.

Command's Structure :

1) LIST
2) LIST name

In the first form, the documents will be displayed one by one, and the user is required to type a semicolon in order for the next document to be displayed. The second form will display only one document.

Example :

LIST invoice ;

Error Messages :

1) In any case of a syntax error, the message
**** LIST - SYNTAX ERROR
will be displayed.

2) If the document, specified in the second form, does not exist,

the message
  DOCUMENT name DOES NOT EXIST
will be displayed.

XREF

XREF is used to display a cross reference of one or more columns in the system, that is the names of all the documents where a certain column can be found. If preceded by PRINT the output will be diverted to the printer.

Command's Structure :

1) XREF
2) XREF name

In the first form all the columns, together with their associated documents, will be displayed one by one and the user is required to type a semicolon in order to display the next column. The second form will display a cross reference for the specified column name.

Example :
  XREF columnl ;

Error Messages :

1) In any case of a syntax error, the message
  **** XREF - SYNTAX ERROR
will be displayed.

2) If the column specified in the second form is not found, the message
  COLUMN NOT FOUND
will be displayed.

ATLIST

ATLIST is used to display the column or document attributes defined in the system. If preceded by a PRINT, the output will be diverted to the printer.

Command's Structure :

1) ATLIST DOCUMENT
2) ATLIST COLUMN
3) ATLIST DOCUMENT name
4) ATLIST COLUMN name

The first and second forms will display the document or column attributes, respectively, one by one, and the user is required to type a semicolon in order for the next attribute to be displayed.

The third and fourth forms will display the specified document or column attribute, respectively.

Examples :

1) ATLIST DOCUMENT ;
2) ATLIST DOCUMENT location ;
3) ATLIST COLUMN type ;

Error Messages :

1) In case of any syntax error, the message
**** ATLIST - SYNTAX ERROR
will be displayed.

2) If the name specified in forms 3 and 4 is not found, the message
name IS NOT FOUND
will be displayed.

SELECT

SELECT is used to display the columns in the system according to the attributes of the associated documents specified by the user. If preceded by PRINT, the output will be diverted to the printer.

Command's Structure :

1) SELECT COLUMN with-without-expressions
2) SELECT with-without-expressions

The with-without-expressions are of the form :
WITH name.name
WITHOUT name.name

The first name is a document attribute, the second is a component of that attribute. The number of WITH or WITHOUT expressions is bounded by 10. This bound can be easily changed by modifying a parameter.

Example :

SELECT COLUMN WITH location.out WITHOUT location.inp ;

Error Messages :

1) In any case of a syntax error, the message
**** SELECT - SYNTAX ERROR
will be displayed.

2) If more than 10 WITH or WITHOUT expressions, the message

WITH / WITHOUT OVERFLOW
will be displayed.

3) If an attribute or a component is not found, the message
ATTRIB/COMPON NOT FOUND
will be displayed.

FIND

FIND is used to display document or column names that may be considered misspelled or similar to a user specified name.

Command's Structure :

1) FIND SIMILAR TO DOCUMENT name
2) FIND DOCUMENT name
3) FIND SIMILAR TO COLUMN name
4) FIND COLUMN name

Forms 1 an 2 are equivalent and so are forms 3 and 4.

Examples :

1) FIND SIMILAR TO DOCUMENT invoice_30 ;
2) FIND COLUMN columnl_name ;

Error Message :

In any case of a syntax error, the message
**** FIND - SYNTAX ERROR
will be displayed.

KEY

KEY is used to declare the key columns in one or more documents, so that later it will be possible to generate functional dependencies.

Command's Structure :

1) KEY
2) KEY name

In the first form, all the documents will be listed one by one, ordered according to the first letter of the document name with the column names. The user will be requested to specify the names of the key columns for every document. In the second form, only the specified document will be listed, and the user will be requested to specify the key columns for that document.

Example :

KEY invoice ;

Error Messages :

1) In any case of a syntax error, the message
**** KEY - SYNTAX ERROR
will be displayed.

2) In case of an illegal symbol or an illegal identifier in the column list the message
ERROR IN COLUMN LIST
will be displayed.

3) If the document specified in the second form is not found, the message
DOCUMENT name DOES NOT EXIST
will be displayed.

4) If a column in the list of columns is not found, the message
COLUMN name NOT FOUND
will be displayed.

PREPARE

PREPARE is used to prepare functional dependencies in a form suitable for input to the Bernstein's algorithm program currently implemented at Kansas State University. The execution is done in two steps. In the first step, abbreviated names are assigned to all the column names that exist in the system. A dictionary of the column names against the abbreviate names is printed at the end of this step. The second step scans all the documents in the system and outputs a set of functional dependencies to a data file.

Command's Structure :

PREPARE

Error Messages :

No error messages.

TEACH

TEACH is used to help the inexperienced user with a summary of the commands supported by the system and a brief description of each command.

Command's Structure :

TEACH

A list of all the commands will be displayed to the user and by typing a name of one of the commands, a brief description will be displayed. By typing END the user will leave the TEACH mode.

Error Messages :

If the user asks for a description of a command that does not exist, the message
WRONG COMMAND NAME
is displayed.

END

END is used to terminate the current session. If a change was made to the data and no SAVE was executed after the change, a message is displayed requesting the user's permission to continue with END. If a NO answer is given, the system returns to normal working mode, if the answer is YES, the session is terminated.

Command's Structure :

END

Error Messages :

No error messages.

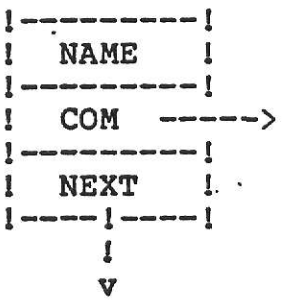
Data Structures

The most trivial data structure is the representation of a document by a single record that contains the document's name and all of the column and attribute names. But using this kind of data structure may cause a) duplication of data since the same column name might appear in more than one document while it is enough to store that name only once; b) Very long response time in the execution of certain commands that need to search all the documents or all the columns. Since the duplication of data and slow responses from the system are to be avoided as well as unnatural restrictions on the data, a different data structure was designed. It might seem in the first reading that the data-structure being used is very clumsy and is not worth the extra code required to maintain it. However understanding the requirements of the system, the need for efficiency and flexibility, the use of PASCAL features and the possibilities to extend the system, as will be described in the fifth section, will lead to an agreement with the chosen data structure.

The Attributes

The system has two kinds of attributes, document attributes and column attributes. Since both kinds have the same characteristics, an identical data structure is used. It is a linked list, where the only difference is in the pointer to the first element of the lists. In the following discussion, the term attribute will refer to either one of the document or the column attributes.

The basic element of the linked list of attributes is the record ATTRIB that has the following structure :



Where NAME is a user defined identifier, COM is a pointer to a list of components and NEXT points to the next attribute in the linked list.

The components of an attribute are represented by a linked list of COMPONENT records. THis record has the following structure :

The field name is the name of the component. Next is a pointer to the next component belonging to the same attribute. BACK is a pointer to the attribute name.

Figure 3.3 illustrates the structure when one atribute, LOCATION, is defined having three components : INP,INT and OUT .

The following formula may be used to calculate the memory requirements of representing an attribute :

(Number_of_Components + 1) * (Name_size + 8)

Using the above formula it is apparent that 144 bytes will be

required to represent the example of figure 3.3 .

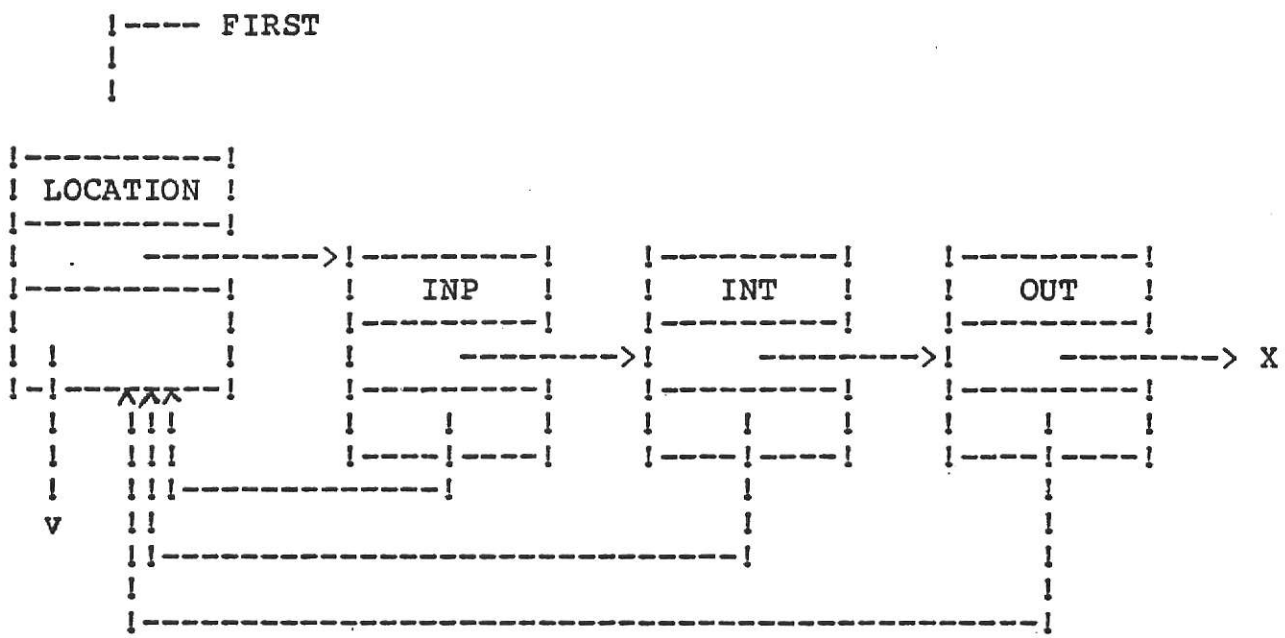
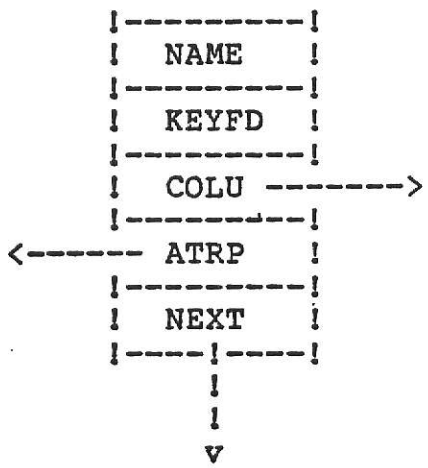


figure 3.3 : Representing the attribute LOCATION

The Documents

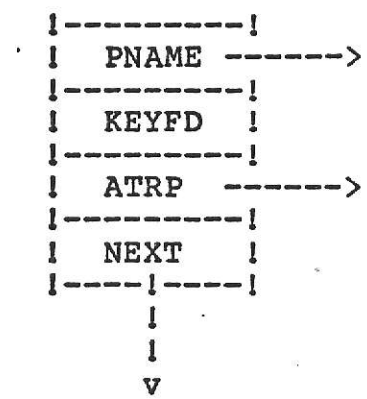
The basic element of the documents structure is the DOCREC record:



The field NAME is a user defined identifier, the document name. THe field KEYFD is a boolean, true or false whether or not a key was declared for this document. COLU is a pointer to a linked list of columns. ATRP is a pointer to a linked list of attributes. NEXT is a pointer to the next record in the list.

Because of considerations of easy access to the documents, the records are not linked together with a simple linked list, where search is sequential and therefore not efficient. Instead, an array, DOCTABLE, with 26 elements corresponding to the letters A to Z, is used. The elements are pointers to DOCREC records, so that 26 linked lists are formed. The access to a certain document is done according to the first character of its name.

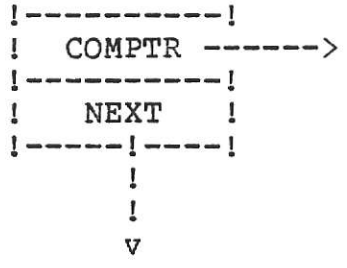
The list of columns, pointed by the field COLU in DOCREC is built of COLREC records :



PNAME is a pointer to a record in the columns table, containing the name of the column. KEYFD is a boolean field that indicates

whether or not this column is a key one. ATRP is a pointer to a linked list of attributes. NEXT points to the next column belonging to the same document.

The attributes lists that are pointed by the fields ATRP in DOCREC and in COLREC are constructed from ATR records.

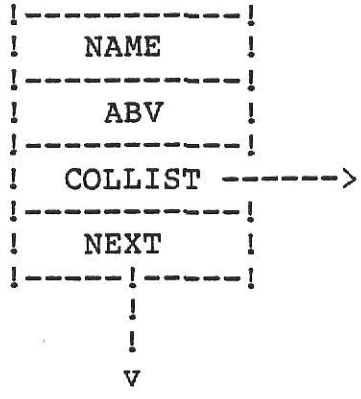


Both of the fields COMPTR and NEXT are pointers. COMPTR points to the appropriate attribute component in the document or column attribute linked lists. NEXT points to the next attribute declared for that document or column.

The Columns

To avoid redundancy and to simplify certain operations, the column names are stored in a distinct data structure.

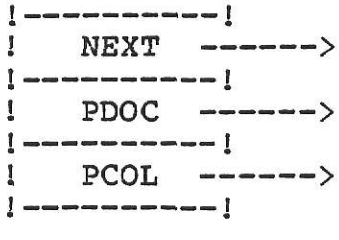
The basic element here is the COLTBREC record that has four fields :



NAME is the column name, ABV is the abbreviated name used for generating functional dependencies as input to the Bernstein's algorithm program. COLLIST is a pointer to a linked list of occurrences of that column. NEXT is a pointer to the next column in the linked list.

Because of reasons similar to the case for documents, an array, COLTABLE, with 26 elements corresponding to the letters A to Z, is used. Each element is a pointer to COLTBREC record. Using this method, 26 linked lists are formed, where all the column names in any one list begin with the same letter.

There is a linked list of occurrences for each column. There is at least one occurrence for every column name, and it is built of COLOCREC records:



PDOC points to the document to which this column belongs. PCOL points to the COLREC record linked to the document pointed by PDOC. NEXT is a pointer to the next occurrence of the column name.

Figure 3.4 illustrates the data structure for one document, "INVOICE", that has the document attribute "LOCATION.INP" and two columns, "COLUMN1" with the column attribute "TYPE.STRING" and "COLUMN2" with the column attribute "TYPE.INTEGER".

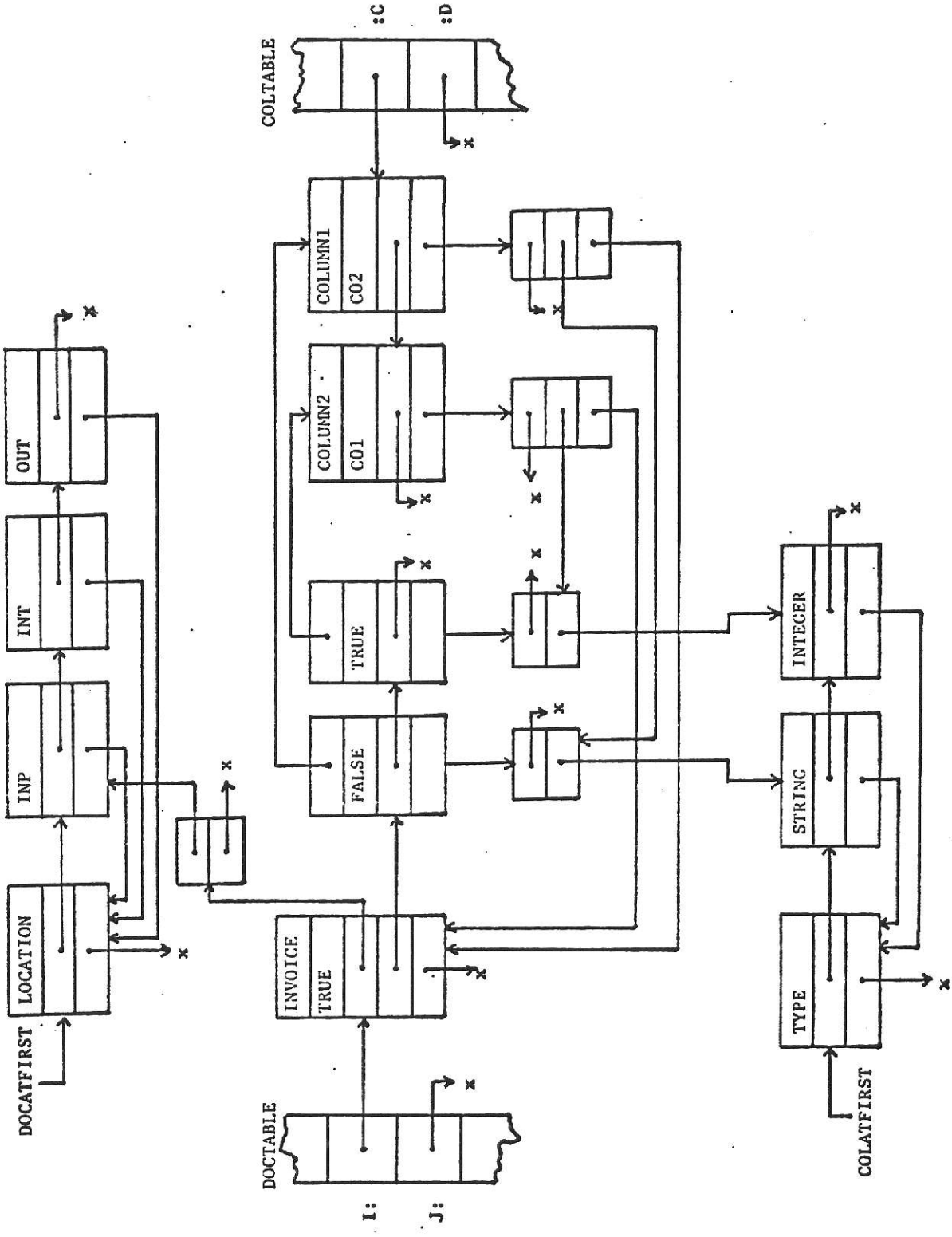


figure 3.4: An Example of the Data Structure Organization

AN EXAMPLE

General Description of the Organization

The organization selected to demonstrate the methodolgy and use of the document handler is a generalized university admission and records sub system, abbreviated UARS for the purpose of this section.

Applications of undergraduate and graduate students are received and processed by the UARS. A decision is made in the UARS whether or not to admit the student to the university. When a student is finally admitted to the university, a student file form is filled with the student's data and is kept together with all the data received by the UARS about that student. In order to be admitted, foreign students are required to fill out a form stating that they are able to furnish a certain amount of money for school every year.

Students who wish to park their car on-campus are required to fill out a vehicle registration form in order to obtain a parking permit. The data on the vehicle registration form is also kept in the student file.

A schedule of classes is published by the UARS for every school semester. Students who wish to enroll, fill an assignment - enrollment form and submit it to the UARS. Adding and dropping courses is done by submitting a change of enrollment form to the UARS. Students may also request to be graded on a pass - fail grade basis by filling out a certain form.
A telephone directory is published once during the school year by the UARS. It includes all students enrolled at the time of publication.

Every instructor receives a class-book for every course that he/she teaches, detailing the students in his/her class and some information about their current level in school. Towards the end of the semester, every instructor receives a class grade form, on which is reported the grades for every student enrolled in the class. Students receive a student grade report by mail with all their grades for the semester and a calculated graduate point average. A student grade history form is kept in the student file with the grades for every course taken. Once the class grade form is submitted, the instructor is able to change a grade only by filling out a grade change form.

The following sections will detail the creation of a third normal form schema from the data collected about the UARS, using the method described in the second section.

Initial Survey

In the initial survey, 16 documents were recorded, with a total of 285 columns. Figure 4.1 is the list of documents provided by the document handler. Figure 4.2 is a list of all the 197 distinct columns found in the system. Figure 4.3 is a complete list of all the documents found in the system, with the specification of their columns and the location attribute (input, resident or output). Figure 4.4 is a complete cross - reference of the columns in the system.

```
DOCUMENT  HANDLER  VER 0.0      RUN UNIVERSITY EXAMPLE STEP0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                    DOCUMENTS IN THE SYSTEM

 1. ASSIGNMENT_ENROLLMENT          WITH    13 COLUMNS
 2. A_PASS_F_GRADING_OPTION        WITH    14 COLUMNS
 3. CLASS_ENROLLMENT               WITH     8 COLUMNS
 4. CHANGE_OF_ENROLLMENT           WITH    16 COLUMNS
 5. CLASS_BOOK                     WITH    13 COLUMNS
 6. CLASS_GRADES                   WITH     4 COLUMNS
 7. FOREIGN_STUDENT_FINANCES       WITH     4 COLUMNS
 8. GRADE_CHANGE                   WITH    11 COLUMNS
 9. GRAD_APPLICATION               WITH    60 COLUMNS
10. SCHEDULE_OF_CLASSES            WITH    10 COLUMNS
11. STUDENT_GRADE_REPORT           WITH    14 COLUMNS
12. STUDENT_GRADE_HISTORY          WITH     6 COLUMNS
13. STUDENT_FILE                   WITH    40 COLUMNS
14. TELEPHONE_DIRECTORY            WITH     8 COLUMNS
15. UNDERGRAD_APPLICATION          WITH    54 COLUMNS
16. VEHICLE_REG_FORM               WITH    10 COLUMNS
```

figure 4.1

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

COLUMNS IN THE SYSTEM

1. ADVISORS_SIGNATURE
2. ADVISOR
3. ADDRESS
4. ATTEN_INSTIT_NAME
5. ATTEN_INSTIT_LOCATION
6. ATTEN_INSTIT_DATES
7. ATTEN_INSTIT_DIPLOMA
8. ATTEN_INSTIT_YEAR_RECEIVED
9. ARMED_FORCES_SERV_DATES
10. APPLICANTS_SIGNATURE
11. APPLICATION_DATE
12. ACCEPTED_TO_DEGREE
13. AGE
14. ATTEN_COLLEGE_NAME
15. ATTEN_COLLEGE_FROM_DATE
16. ATTEN_COLLEGE_TO_DATE
17. ACT_DATE
18. ADDRESS_STREET
19. ADDRESS_CITY
20. ADDRESS_STATE
21. ADDRESS_ZIP
22. COLLEGE_GRAD_SCHOOL
23. COURSE_NUMBER
24. COURSE_TITLE
25. CREDIT_HOURS
26. CREDIT_STATUS
27. COLLEGE_CURRICULUM
28. COURSE_NAME
29. COUNTRY_OF_CITIZENSHIP
30. CREDIT_NO_CREDIT
31. CURRICULUM
32. CURRICULUM_NUMBER
33. DEANS_SIGNATURE
34. DEL_SEL_A_PASS_F_OPTION
35. DROP_ADD_CHANGE
36. DEANS_OFFICE
37. DATE
38. DEGREE
39. DATE_OF_KANSAS_RESIDENT
40. DATE_OF_BIRTH
41. DAYS
42. DATE_ENROLLED
43. DATE_PROJ_GRAD
44. DATE_OF_KANSAS_RESIDENCY
45. DATE_OF_PARENTS_KS_RESIDENCY
46. DATES_OF_MILITARY_SERVICE
47. DEPENDENT_OF_MILITARY
48. DEGREE_SELECTION
49. DATE_ATTEN_KSU
50. DATE_OF_APPLICATION
51. EFFECTIVE_DATE
52. ETHNIC_RACIAL_STATUS
53. EVER_BEEN_IN_COLLEGE

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

54. ENTER_SEMESTER
55. ENTER_YEAR
56. FOREIGN_ST_SUPPORT_SCHOLAR
57. FOREIGN_ST_SUPPORT_LOAN
58. FOREIGN_ST_SUPPORT_PERSONAL
59. FOREIGN_ST_SUPPORT_SUMMER_EX
60. FROM_HOUR
61. GRADE
62. GRADUATED_NAME
63. GRADUATED_CITY
64. GRADUATED_STATE
65. GRADUATED_DATE
66. HOME_ADDRESS_STREET
67. HOME_ADDRESS_CITY
68. HOME_ADDRESS_COUNTY
69. HOME_ADDRESS_STATE
70. HOME_ADDRESS_ZIP
71. HIGH_SCHOOL_NAME
72. HIGH_SCHOOL_CITY
73. HIGH_SCHOOL_STATE
74. HIGH_SCHOOL_DATE_FROM
75. HIGH_SCHOOL_DATE_TO
76. INSTRUCTORS_SIGNATURE
77. INSTRUCTOR
78. LINE_NUMBER
79. LOCAL_ADDRESS_STREET
80. LOCAL_ADDRESS_CITY
81. LOCAL_ADDRESS_STATE
82. LOCAL_ADDRESS_ZIP
83. MEETING_PLACE
84. MEETING_TIME
85. MEETING_DAYS
86. MAJOR
87. NAME_MIDDLE
88. MARITAL_STATUS
89. NAME
90. NEW_GRADE
91. NAME_LAST
92. NAME_FIRST
93. NAME_MIDDLE
94. NOW_IN_COLLEGE
95. OTHER_NAMES
96. OTHER_LAST_NAME
97. PREV_GRADE
98. PHONE_NUMBER
99. PERMANENT_ADDRESS_STREET
100. PERMANENT_ADDRESS_CITY
101. PERMANENT_ADDRESS_COUNTY
102. PERMANENT_ADDRESS_STATE
103. PERMANENT_ADDRESS_ZIP
104. PRESENT_ADDRESS_STREET
105. PRESENT_ADDRESS_CITY
106. PRESENT_ADDRESS_STATE
107. PRESENT_ADDRESS_ZIP
108. PLACE_OF_BIRTH

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

109. PERSON_IN_EMERGENCY_LAST_NA
110. PERSON_IN_EMERGENCY_FIRST_NA
111. PERSON_IN_EMERGENCY_MID_NA
112. PERSON_IN_EMER_STREET
113. PERSON_IN_EMER_CITY
114. PERSON_IN_EMER_COUNTY
115. PERSON_IN_EMER_STATE
116. POSITION_HELD_NATURE
117. POSITION_HELD_COMPANY
118. POSITION_HELD_LOCATION
119. POSITION_HELD_DATES
120. PLACE_OF_BIRTH_CITY
121. PLACE_OF_BIRTH_STATE
122. PARENT_NAME
123. PARENT_STREET
124. PARENT_CITY
125. PARENT_STATE
126. PARENT_ZIP
127. PARENT_PHONE
128. PLAN_TO_COMPLETE_IN_KSU
129. PREV_ATTEN_KSU
130. REF_INSTR_NAME
131. REF_INSTR_POSITION
132. REF_INSTR_ADDRESS
133. ROOM_NUMBER
134. RESIDENT_GPA
135. RELATIVE_NAME
136. RELATIVE_TELEPHONE
137. RELATIVE_ADDRESS_STREET
138. RELATIVE_ADDRESS_CITY
139. RELATIVE_ADDRESS_STATE
140. RELATIVE_ADDRESS_ZIP
141. RELATIVE_ATTEN_KSU
142. RELATIVE_ATTEN_KSU_DATES
143. RELIGIOUS_PREFERENCE
144. SOCIAL_SECURITY_NUMBER
145. SEMESTER
146. STUDENT_SIGNATURE
147. SUBMISSION_DEADLINE
148. STUDENT_NAME
149. SCH
150. S_CL
151. SIGNATURE
152. SEMESTER_OF_ADMISSION
153. SEX
154. SCHOL_HONOR_PRIZES
155. SPEC_STUD_SIGNATURE
156. SPEC_STUD_APPLICATION_DATE
157. STUDENT_ADDRESS_STREET
158. STUDENT_ADDRESS_CITY
159. STUDENT_ADDRESS_STATE
160. STUDENT_ADDRESS_ZIP
161. STUDENT_PERM_AD_STREET
162. STUDENT_PERM_AD_CITY
163. STUDENT_PERM_AD_COUNTY

164. STUDENT_PERM_AD_STATE
165. STUDENT_PERM_AD_ZIP
166. STATUS_COLLEGE
167. STATUS_LEVEL
168. STATUS_PRIMARY_MAJOR
169. STATUS_SECONDARY_MAJOR
170. STATUS_ADVISOR
171. STATUS_DEGREE_SOUGHT
172. STUDENT_PHONE
173. STUDENT_CLASS_ID
174. STUDENT_SCHOOL
175. SOCIAL_SECURITY
176. TODAYS_DATE
177. TYPE_OF_ADMISSION
178. TOTAL_IN_STATE_FEES
179. TOTAL_OUT_STATE_FEES
180. TO_HOUR
181. TOTAL_CR_HOURS
182. TRANSFER_GPA
183. TELEPHONE
184. TYPE_OF_PREV_PROGRAM
185. UND_GRAD_CREDIT
186. US_CITIZEN
187. VISA_TYPE
188. VEHICLE_STATE
189. VEHICLE_LICENSE_NUMBER
190. VEHICLE_MAKE
191. VEHICLE_YEAR
192. VEHICLE_PERMIT_NUMBER
193. WITHDRAW_FROM_UNIVERSITY
194. WAIVE_RIGHT_OF_ACCESS
195. YEAR
196. YOUR_COLLEGE
197. YEAR_OF_ADMISSION

figure 4.2

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

DOCUMENT / COLUMN LISTING

ASSIGNMENT_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

NAME
COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE
LINE_NUMBER
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
CREDIT_STATUS
ADVISORS_SIGNATURE
DEANS_SIGNATURE
SOCIAL_SECURITY_NUMBER
SEMESTER
YEAR

A_PASS_F_GRADING_OPTION

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

NAME
YOUR_COLLEGE
TODAYS_DATE
LINE_NUMBER
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
DEANS_SIGNATURE
ADVISORS_SIGNATURE
STUDENT_SIGNATURE
SOCIAL_SECURITY_NUMBER
SEMESTER
YEAR
DEL_SEL_A_PASS_F_OPTION

CLASS_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

SOCIAL_SECURITY_NUMBER
NAME
COLLEGE_CURRICULUM
SUBMISSION_DEADLINE
ADVISOR
CREDIT_STATUS
CREDIT_HOURS
COURSE_NUMBER

CHANGE_OF_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

NAME
SOCIAL_SECURITY_NUMBER
COLLEGE_GRAD_SCHOOL

EFFECTIVE_DATE
LINE_NUMBER
DROP_ADD_CHANGE
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
UND_GRAD_CREDIT
ADVISORS_SIGNATURE
DEANS_OFFICE
STUDENT_SIGNATURE
SEMESTER
YEAR
WITHDRAW_FROM_UNIVERSITY

CLASS_BOOK

DOCUMENT ATTRIBUTES :
LOCATION . OUTPUT

LINE_NUMBER
COURSE_NUMBER
COURSE_NAME
SEMESTER
MEETING_PLACE
SOCIAL_SECURITY_NUMBER
STUDENT_NAME
CREDIT_STATUS
CREDIT_HOURS
SCH
S_CL
MEETING_TIME
MEETING_DAYS

CLASS_GRADES

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

COURSE_NUMBER
SOCIAL_SECURITY_NUMBER
GRADE
INSTRUCTORS_SIGNATURE

FOREIGN_STUDENT_FINANCES

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

NAME
ADDRESS
SIGNATURE
DATE

GRADE_CHANGE

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

STUDENT_NAME
SOCIAL_SECURITY_NUMBER
PREV_GRADE
CREDIT_HOURS
COURSE_NAME

```
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
                    NEW_GRADE
                    DATE
                    INSTRUCTOR

GRAD_APPLICATION
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . INPUT
                    SEMESTER_OF_ADMISSION
                    YEAR_OF_ADMISSION
                    DEGREE
                    MAJOR
                    SOCIAL_SECURITY_NUMBER
                    SEX
                    PHONE_NUMBER
                    NAME_LAST
                    NAME_FIRST
                    NAME_MIDDLE
                    OTHER_NAMES
                    PERMANENT_ADDRESS_STREET
                    PERMANENT_ADDRESS_CITY
                    PERMANENT_ADDRESS_COUNTY
                    PERMANENT_ADDRESS_STATE
                    PERMANENT_ADDRESS_ZIP
                    PRESENT_ADDRESS_STREET
                    PRESENT_ADDRESS_CITY
                    PRESENT_ADDRESS_STATE
                    PRESENT_ADDRESS_ZIP
                    DATE_OF_KANSAS_RESIDENT
                    PLACE_OF_BIRTH
                    DATE_OF_BIRTH
                    COUNTRY_OF_CITIZENSHIP
                    VISA_TYPE
                    PERSON_IN_EMERGENCY_LAST_NA
                    PERSON_IN_EMERGENCY_FIRST_NA
                    PERSON_IN_EMERGENCY_MID_NA
                    PERSON_IN_EMER_STREET
                    PERSON_IN_EMER_CITY
                    PERSON_IN_EMER_COUNTY
                    PERSON_IN_EMER_STATE
                    ATTEN_INSTIT_NAME
                    ATTEN_INSTIT_LOCATION
                    ATTEN_INSTIT_DATES
                    ATTEN_INSTIT_DIPLOMA
                    ATTEN_INSTIT_YEAR_RECEIVED
                    POSITION_HELD_NATURE
                    POSITION_HELD_COMPANY
                    POSITION_HELD_LOCATION
                    POSITION_HELD_DATES
                    ARMED_FORCES_SERV_DATES
                    REF_INSTR_NAME
                    REF_INSTR_POSITION
                    REF_INSTR_ADDRESS
```

```
                    WAIVE_RIGHT_OF_ACCESS
                    SCHOL_HONOR_PRIZES
                    ETHNIC_RACIAL_STATUS
                    APPLICANTS_SIGNATURE
                    APPLICATION_DATE
                    SPEC_STUD_SIGNATURE
                    SPEC_STUD_APPLICATION_DATE
                    ACCEPTED_TO_DEGREE
                    TYPE_OF_ADMISSION
                    FOREIGN_ST_SUPPORT_SCHOLAR
                    FOREIGN_ST_SUPPORT_LOAN
                    FOREIGN_ST_SUPPORT_PERSONAL
                    FOREIGN_ST_SUPPORT_SUMMER_EX
                    TOTAL_IN_STATE_FEES
                    TOTAL_OUT_STATE_FEES

SCHEDULE_OF_CLASSES
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                    LINE_NUMBER
                    COURSE_NUMBER
                    COURSE_TITLE
                    CREDIT_NO_CREDIT
                    CREDIT_HOURS
                    DAYS
                    FROM_HOUR
                    TO_HOUR
                    ROOM_NUMBER
                    INSTRUCTOR

STUDENT_GRADE_REPORT
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . OUTPUT
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_NAME
                    STUDENT_ADDRESS_STREET
                    STUDENT_ADDRESS_CITY
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
                    CREDIT_HOURS
                    GRADE
                    TOTAL_CR_HOURS
                    RESIDENT_GPA
                    TRAANSFER_GPA

STUDENT_GRADE_HISTORY
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                    SOCIAL_SECURITY_NUMBER
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
```

CREDIT_HOURS
GRADE

STUDENT_FILE

DOCUMENT ATTRIBUTES :
LOCATION . RESIDENT

SOCIAL_SECURITY_NUMBER
STUDENT_NAME
SEX
STUDENT_PERM_AD_STREET
STUDENT_PERM_AD_CITY
STUDENT_PERM_AD_COUNTY
STUDENT_PERM_AD_STATE
STUDENT_PERM_AD_ZIP
PLACE_OF_BIRTH_CITY
PLACE_OF_BIRTH_STATE
DATE_OF_BIRTH
GRADUATED_NAME
GRADUATED_CITY
GRADUATED_STATE
GRADUATED_DATE
PARENT_NAME
PARENT_STREET
PARENT_CITY
PARENT_STATE
PARENT_ZIP
PARENT_PHONE
DATE_ENROLLED
DATE_PROJ_GRAD
STATUS_COLLEGE
STATUS_LEVEL
STATUS_PRIMARY_MAJOR
STATUS_SECONDARY_MAJOR
STATUS_ADVISOR
STATUS_DEGREE_SOUGHT
LOCAL_ADDRESS_STREET
LOCAL_ADDRESS_CITY
LOCAL_ADDRESS_STATE
LOCAL_ADDRESS_ZIP
TELEPHONE
MARITAL_STATUS
VEHICLE_STATE
VEHICLE_LICENSE_NUMBER
VEHICLE_MAKE
VEHICLE_YEAR
VEHICLE_PERMIT_NUMBER

TELEPHONE_DIRECTORY

DOCUMENT ATTRIBUTES :
LOCATION . OUTPUT

STUDENT_NAME
STUDENT_ADDRESS_STREET
STUDENT_PHONE
STUDENT_CLASS_ID
STUDENT_SCHOOL

STUDENT_PERM_AD_STREET
STUDENT_PERM_AD_CITY
STUDENT_PERM_AD_STATE

UNDERGRAD_APPLICATION

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

NAME_LAST
NAME_FIRST
NAME_MIDDLE
TELEPHONE
OTHER_LAST_NAME
HOME_ADDRESS_STREET
HOME_ADDRESS_CITY
HOME_ADDRESS_COUNTY
HOME_ADDRESS_STATE
HOME_ADDRESS_ZIP
DATE_OF_BIRTH
AGE
PLACE_OF_BIRTH_CITY
PLACE_OF_BIRTH_STATE
SEX
SOCIAL_SECURITY
US_CITIZEN
VISA_TYPE
RELATIVE_NAME
RELATIVE_TELEPHONE
RELATIVE_ADDRESS_STREET
RELATIVE_ADDRESS_CITY
RELATIVE_ADDRESS_SATE
RELATIVE_ADDRESS_ZIP
DATE_OF_KANSAS_RESIDENCY
DATE_OF_PARENTS_KS_RESIDENCY
DATES_OF_MILITARY_SERVICE
DEPENDENT_OF_MILITARY
HIGH_SCHOOL_NAME
HIGH_SCHOOL_CITY
HIGH_SCHOOL_STATE
HIGH_SCHOOL_DATE_FROM
HIGH_SCHOOL_DATE_TO
NOW_IN_COLLEGE
EVER_BEEN_IN_COLLEGE
ATTEN_COLLEGE_NAME
ATTEN_COLLEGE_FROM_DATE
ATTEN_COLLEGE_TO_DATE
ACT_DATE
CURRICULUM
CURRICULUM_NUMBER
DEGREE_SELECTION
PLAN_TO_COMPLETE_IN_KSU
PREV_ATTEN_KSU
DATE_ATTEN_KSU
TYPE_OF_PREV_PROGRAM
RELATIVE_ATTEN_KSU
RELATIVE_ATTEN_KSU_DATES

ENTER_SEMESTER
ENTER_YEAR
ETHNIC_RACIAL_STATUS
RELIGIOUS_PREFERENCE
APPLICANTS_SIGNATURE
DATE_OF_APPLICATION

VEHICLE_REG_FORM

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

VEHICLE_STATE
VEHICLE_LICENSE_NUMBER
VEHICLE_MAKE
VEHICLE_YEAR
VEHICLE_PERMIT_NUMBER
SOCIAL_SECURITY_NUMBER
ADDRESS_STREET
ADDRESS_CITY
ADDRESS_STATE
ADDRESS_ZIP

figure 4.3

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

COLUMN CROSS REFERENCE

ADVISORS_SIGNATURE	ASSIGNMENT_ENROLLMENT A_PASS_F_GRADING_OPTION CHANGE_OF_ENROLLMENT
ADVISOR	CLASS_ENROLLMENT
ADDRESS	FOREGIN_STUDENT_FINANCES
ATTEN_INSTIT_NAME	GRAD_APPLICATION
ATTEN_INSTIT_LOCATION	GRAD_APPLICATION
ATTEN_INSTIT_DATES	GRAD_APPLICATION
ATTEN_INSTIT_DIPLOMA	GRAD_APPLICATION
ATTEN_INSTIT_YEAR_RECEIVED	GRAD_APPLICATION
ARMED_FORCES_SERV_DATES	GRAD_APPLICATION
APPLICANTS_SIGNATURE	GRAD_APPLICATION UNDERGRAD_APPLICATION
APPLICATION_DATE	GRAD_APPLICATION
ACCEPTED_TO_DEGREE	GRAD_APPLICATION
AGE	UNDERGRAD_APPLICATION
ATTEN_COLLEGE_NAME	UNDERGRAD_APPLICATION
ATTEN_COLLEGE_FROM_DATE	UNDERGRAD_APPLICATION
ATTEN_COLLEGE_TO_DATE	UNDERGRAD_APPLICATION
ACT_DATE	UNDERGRAD_APPLICATION
ADDRESS_STREET	VEHICLE_REG_FORM
ADDRESS_CITY	VEHICLE_REG_FORM
ADDRESS_STATE	VEHICLE_REG_FORM
ADDRESS_ZIP	VEHICLE_REG_FORM
COLLEGE_GRAD_SCHOOL	ASSIGNMENT_ENROLLMENT CHANGE_OF_ENROLLMENT
COURSE_NUMBER	ASSIGNMENT_ENROLLMENT A_PASS_F_GRADING_OPTION CLASS_ENROLLMENT CHANGE_OF_ENROLLMENT

	CLASS_BOOK
	CLASS_GRADES
	GRADE_CHANGE
	SCHEDULE_OF_CLASSES
	STUDENT_GRADE_REPORT
	STUDENT_GRADE_HISTORY
COURSE_TITLE	
	ASSIGNMENT_ENROLLMENT
	A_PASS_F_GRADING_OPTION
	CHANGE_OF_ENROLLMENT
	SCHEDULE_OF_CLASSES
CREDIT_HOURS	
	ASSIGNMENT_ENROLLMENT
	A_PASS_F_GRADING_OPTION
	CLASS_ENROLLMENT
	CHANGE_OF_ENROLLMENT
	CLASS_BOOK
	GRADE_CHANGE
	SCHEDULE_OF_CLASSES
	STUDENT_GRADE_REPORT
	STUDENT_GRADE_HISTORY
CREDIT_STATUS	
	ASSIGNMENT_ENROLLMENT
	CLASS_ENROLLMENT
	CLASS_BOOK
COLLEGE_CURRICULUM	
	CLASS_ENROLLMENT
COURSE_NAME	
	CLASS_BOOK
	GRADE_CHANGE
COUNTRY_OF_CITIZENSHIP	
	GRAD_APPLICATION
CREDIT_NO_CREDIT	
	SCHEDULE_OF_CLASSES
CURRICULUM	
	UNDERGRAD_APPLICATION
CURRICULUM_NUMBER	
	UNDERGRAD_APPLICATION
DEANS_SIGNATURE	
	ASSIGNMENT_ENROLLMENT
	A_PASS_F_GRADING_OPTION
DEL_SEL_A_PASS_F_OPTION	
	A_PASS_F_GRADING_OPTION
DROP_ADD_CHANGE	
	CHANGE_OF_ENROLLMENT
DEANS_OFFICE	
	CHANGE_OF_ENROLLMENT
DATE	
	FOREIGN_STUDENT_FINANCES
	GRADE_CHANGE
DEGREE	
	GRAD_APPLICATION
DATE_OF_KANSAS_RESIDENT	
	GRAD_APPLICATION
DATE_OF_BIRTH	

	GRAD_APPLICATION
	STUDENT_FILE
	UNDERGRAD_APPLICATION
DAYS	
	SCHEDULE_OF_CLASSES
DATE_ENROLLED	
	STUDENT_FILE
DATE_PROJ_GRAD	
	STUDENT_FILE
DATE_OF_KANSAS_RESIDENCY	
	UNDERGRAD_APPLICATION
DATE_OF_PARENTS_KS_RESIDENCY	
	UNDERGRAD_APPLICATION
DATES_OF_MILITARY_SERVICE	
	UNDERGRAD_APPLICATION
DEPENDENT_OF_MILITARY	
	UNDERGRAD_APPLICATION
DEGREE_SELECTION	
	UNDERGRAD_APPLICATION
DATE_ATTEN_KSU	
	UNDERGRAD_APPLICATION
DATE_OF_APPLICATION	
	UNDERGRAD_APPLICATION
EFFECTIVE_DATE	
	ASSIGNMENT_ENROLLMENT
	CHANGE_OF_ENROLLMENT
ETHNIC_RACIAL_STATUS	
	GRAD_APPLICATION
	UNDERGRAD_APPLICATION
EVER_BEEN_IN_COLLEGE	
	UNDERGRAD_APPLICATION
ENTER_SEMESTER	
	UNDERGRAD_APPLICATION
ENTER_YEAR	
	UNDERGRAD_APPLICATION
FOREIGN_ST_SUPPORT_SCHOLAR	
	GRAD_APPLICATION
FOREIGN_ST_SUPPORT_LOAN	
	GRAD_APPLICATION
FOREIGN_ST_SUPPORT_PERSONAL	
	GRAD_APPLICATION
FOREIGN_ST_SUPPORT_SUMMER_EX	
	GRAD_APPLICATION
FROM_HOUR	
	SCHEDULE_OF_CLASSES
GRADE	
	CLASS_GRADES
	STUDENT_GRADE_REPORT
	STUDENT_GRADE_HISTORY
GRADUATED_NAME	
	STUDENT_FILE
GRADUATED_CITY	
	STUDENT_FILE
GRADUATED_STATE	
	STUDENT_FILE

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

GRADUATED_DATE	STUDENT_FILE
HOME_ADDRESS_STREET	UNDERGRAD_APPLICATION
HOME_ADDRESS_CITY	UNDERGRAD_APPLICATION
HOME_ADDRESS_COUNTY	UNDERGRAD_APPLICATION
HOME_ADDRESS_STATE	UNDERGRAD_APPLICATION
HOME_ADDRESS_ZIP	UNDERGRAD_APPLICATION
HIGH_SCHOOL_NAME	UNDERGRAD_APPLICATION
HIGH_SCHOOL_CITY	UNDERGRAD_APPLICATION
HIGH_SCHOOL_STATE	UNDERGRAD_APPLICATION
HIGH_SCHOOL_DATE_FROM	UNDERGRAD_APPLICATION
HIGH_SCHOOL_DATE_TO	UNDERGRAD_APPLICATION
INSTRUCTORS_SIGNATURE	CLASS_GRADES
INSTRUCTOR	GRADE_CHANGE SCHEDULE_OF_CLASSES
LINE_NUMBER	ASSIGNMENT_ENROLLMENT A_PASS_F_GRADING_OPTION CHANGE_OF_ENROLLMENT CLASS_BOOK SCHEDULE_OF_CLASSES
LOCAL_ADDRESS_STREET	STUDENT_FILE
LOCAL_ADDRESS_CITY	STUDENT_FILE
LOCAL_ADDRESS_STATE	STUDENT_FILE
LOCAL_ADDRESS_ZIP	STUDENT_FILE
MEETING_PLACE	CLASS_BOOK
MEETING_TIME	CLASS_BOOK
MEETING_DAYS	CLASS_BOOK
MAJOR	GRAD_APPLICATION
NAME_MIDDLE	GRAD_APPLICATION
MARITAL_STATUS	STUDENT_FILE
NAME	ASSIGNMENT_ENROLLMENT

	A_PASS_F_GRADING_OPTION
	CLASS_ENROLLMENT
	CHANGE_OF_ENROLLMENT
	FOREIGN_STUDENT_FINANCES
NEW_GRADE	
	GRADE_CHANGE
NAME_LAST	
	GRAD_APPLICATION
	UNDERGRAD_APPLICATION
NAME_FIRST	
	GRAD_APPLICATION
	UNDERGRAD_APPLICATION
NAME_MIDDLE	
	UNDERGRAD_APPLICATION
NOW_IN_COLLEGE	
	UNDERGRAD_APPLICATION
OTHER_NAMES	
	GRAD_APPLICATION
OTHER_LAST_NAME	
	UNDERGRAD_APPLICATION
PREV_GRADE	
	GRADE_CHANGE
PHONE_NUMBER	
	GRAD_APPLICATION
PERMANENT_ADDRESS_STREET	
	GRAD_APPLICATION
PERMANENT_ADDRESS_CITY	
	GRAD_APPLICATION
PERMANENT_ADDRESS_COUNTY	
	GRAD_APPLICATION
PERMANENT_ADDRESS_STATE	
	GRAD_APPLICATION
PERMANENT_ADDRESS_ZIP	
	GRAD_APPLICATION
PRESENT_ADDRESS_STREET	
	GRAD_APPLICATION
PRESENT_ADDRESS_CITY	
	GRAD_APPLICATION
PRESENT_ADDRESS_STATE	
	GRAD_APPLICATION
PRESENT_ADDRESS_ZIP	
	GRAD_APPLICATION
PLACE_OF_BIRTH	
	GRAD_APPLICATION
PERSON_IN_EMERGENCY_LAST_NA	
	GRAD_APPLICATION
PERSON_IN_EMERGENCY_FIRST_NA	
	GRAD_APPLICATION
PERSON_IN_EMERGENCY_MID_NA	
	GRAD_APPLICATION
PERSON_IN_EMER_STREET	
	GRAD_APPLICATION
PERSON_IN_EMER_CITY	
	GRAD_APPLICATION
PERSON_IN_EMER_COUNTY	

	GRAD_APPLICATION
PERSON_IN_EMER_STATE	GRAD_APPLICATION
POSITION_HELD_NATURE	GRAD_APPLICATION
POSITION_HELD_COMPANY	GRAD_APPLICATION
POSITION_HELD_LOCATION	GRAD_APPLICATION
POSITION_HELD_DATES	GRAD_APPLICATION
PLACE_OF_BIRTH_CITY	STUDENT_FILE UNDERGRAD_APPLICATION
PLACE_OF_BIRTH_STATE	STUDENT_FILE UNDERGRAD_APPLICATION
PARENT_NAME	STUDENT_FILE
PARENT_STREET	STUDENT_FILE
PARENT_CITY	STUDENT_FILE
PARENT_STATE	STUDENT_FILE
PARENT_ZIP	STUDENT_FILE
PARENT_PHONE	STUDENT_FILE
PLAN_TO_COMPLETE_IN_KSU	UNDERGRAD_APPLICATION
PREV_ATTEN_KSU	UNDERGRAD_APPLICATION
REF_INSTR_NAME	GRAD_APPLICATION
REF_INSTR_POSITION	GRAD_APPLICATION
REF_INSTR_ADDRESS	GRAD_APPLICATION
ROOM_NUMBER	SCHEDULE_OF_CLASSES
RESIDENT_GPA	STUDENT_GRADE_REPORT
RELATIVE_NAME	UNDERGRAD_APPLICATION
RELATIVE_TELEPHONE	UNDERGRAD_APPLICATION
RELATIVE_ADDRESS_STREET	UNDERGRAD_APPLICATION
RELATIVE_ADDRESS_CITY	UNDERGRAD_APPLICATION
RELATIVE_ADDRESS_STATE	UNDERGRAD_APPLICATION
RELATIVE_ADDRESS_ZIP	UNDERGRAD_APPLICATION

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

RELATIVE_ATTEN_KSU	UNDERGRAD_APPLICATION
RELATIVE_ATTEN_KSU_DATES	UNDERGRAD_APPLICATION
RELIGIOUS_PREFERENCE	UNDERGRAD_APPLICATION
SOCIAL_SECURITY_NUMBER	ASSIGNMENT_ENROLLMENT A_PASS_F_GRADING_OPTION CLASS_ENROLLMENT CHANGE_OF_ENROLLMENT CLASS_BOOK CLASS_GRADES GRADE_CHANGE GRAD_APPLICATION STUDENT_GRADE_REPORT STUDENT_GRADE_HISTORY STUDENT_FILE VEHICLE_REG_FORM
SEMESTER	ASSIGNMENT_ENROLLMENT A_PASS_F_GRADING_OPTION CHANGE_OF_ENROLLMENT CLASS_BOOK GRADE_CHANGE STUDENT_GRADE_REPORT STUDENT_GRADE_HISTORY
STUDENT_SIGNATURE	A_PASS_F_GRADING_OPTION CHANGE_OF_ENROLLMENT
SUBMISSION_DEADLINE	CLASS_ENROLLMENT
STUDENT_NAME	CLASS_BOOK GRADE_CHANGE STUDENT_GRADE_REPORT STUDENT_FILE TELEPHONE_DIRECTORY
SCH	CLASS_BOOK
S_CL	CLASS_BOOK
SIGNATURE	FOREIGN_STUDENT_FINANCES
SEMESTER_OF_ADMISSION	GRAD_APPLICATION
SEX	GRAD_APPLICATION STUDENT_FILE UNDERGRAD_APPLICATION
SCHOL_HONOR_PRIZES	GRAD_APPLICATION
SPEC_STUD_SIGNATURE	GRAD_APPLICATION
SPEC_STUD_APPLICATION_DATE	

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

	GRAD_APPLICATION
STUDENT_ADDRESS_STREET	
	STUDENT_GRADE_REPORT TELEPHONE_DIRECTORY
STUDENT_ADDRESS_CITY	
	STUDENT_GRADE_REPORT
STUDENT_ADDRESS_STATE	
	STUDENT_GRADE_REPORT
STUDENT_ADDRESS_ZIP	
	STUDENT_GRADE_REPORT
STUDENT_PERM_AD_STREET	
	STUDENT_FILE TELEPHONE_DIRECTORY
STUDENT_PERM_AD_CITY	
	STUDENT_FILE TELEPHONE_DIRECTORY
STUDENT_PERM_AD_COUNTY	
	STUDENT_FILE
STUDENT_PERM_AD_STATE	
	STUDENT_FILE TELEPHONE_DIRECTORY
STUDENT_PERM_AD_ZIP	
	STUDENT_FILE
STATUS_COLLEGE	
	STUDENT_FILE
STATUS_LEVEL	
	STUDENT_FILE
STATUS_PRIMARY_MAJOR	
	STUDENT_FILE
STATUS_SECONDARY_MAJOR	
	STUDENT_FILE
STATUS_ADVISOR	
	STUDENT_FILE
STATUS_DEGREE_SOUGHT	
	STUDENT_FILE
STUDENT_PHONE	
	TELEPHONE_DIRECTORY
STUDENT_CLASS_ID	
	TELEPHONE_DIRECTORY
STUDENT_SCHOOL	
	TELEPHONE_DIRECTORY
SOCIAL_SECURITY	
	UNDERGRAD_APPLICATION
TODAYS_DATE	
	A_PASS_F_GRADING_OPTION
TYPE_OF_ADMISSION	
	GRAD_APPLICATION
TOTAL_IN_STATE_FEES	
	GRAD_APPLICATION
TOTAL_OUT_STATE_FEES	
	GRAD_APPLICATION
TO_HOUR	
	SCHEDULE_OF_CLASSES
TOTAL_CR_HOURS	
	STUDENT_GRADE_REPORT

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP0

TRANSFER_GPA	STUDENT_GRADE_REPORT
TELEPHONE	STUDENT_FILE UNDERGRAD_APPLICATION
TYPE_OF_PREV_PROGRAM	UNDERGRAD_APPLICATION
UND_GRAD_CREDIT	CHANGE_OF_ENROLLMENT
US_CITIZEN	UNDERGRAD_APPLICATION
VISA_TYPE	GRAD_APPLICATION UNDERGRAD_APPLICATION
VEHICLE_STATE	STUDENT_FILE VEHICLE_REG_FORM
VEHICLE_LICENSE_NUMBER	STUDENT_FILE VEHICLE_REG_FORM
VEHICLE_MAKE	STUDENT_FILE VEHICLE_REG_FORM
VEHICLE_YEAR	STUDENT_FILE VEHICLE_REG_FORM
VEHICLE_PERMIT_NUMBER	STUDENT_FILE VEHICLE_REG_FORM
WITHDRAW_FROM_UNIVERSITY	CHANGE_OF_ENROLLMENT
WAIVE_RIGHT_OF_ACCESS	GRAD_APPLICATION
YEAR	ASSIGNMENT_ENROLLMENT A_PASS_F_GRADING_OPTION CHANGE_OF_ENROLLMENT GRADE_CHANGE STUDENT_GRADE_REPORT STUDENT_GRADE_HISTORY
YOUR_COLLEGE	A_PASS_F_GRADING_OPTION
YEAR_OF_ADMISSION	GRAD_APPLICATION

figure 4.4

Deletion of Synonyms

Once the organization survey is finished and the collection of documents is done, it is time to start with the one-by-one analysis of the documents and the columns found in them.

Table 4.1 is a list of all the synonyms among the columns, found in the UARS documents.

An example of a simple synonym is the column EFFECTIVE_DATE found in several documents. This has the same meaning as the column TODAYS_DATE in the document A_PASS_F_GRADING_OPTION.

It is sometimes necessary to specify as synonyms and thus merge more than one column in a certain document. For example, the columns ENTER_SEMESTER and ENTER_YEAR in the document UNDERGRAD_APPLICATION were merged into the column DATE_ENROLLED. On the other hand, it is sometimes necessary to split a column into two or more columns. For example, the column ADDRESS in the document FOREIGN_STUDENT_FINANCES was split into the columns STUDENT_PERM_AD_STREET, STUDENT_PERM_AD_CITY, STUDENT_PERM_AD_COUNTY, STUDENT_PERM_AD_STATE and STUDENT_PERM_AD_ZIP.

Figure 4.5 is a complete list of the UARS documents as they are represented by the document handler after the synonym columns were renamed. The total number of columns now is 279, a decrease of 6 columns as compared to the initial number. The number of distinct columns is now 129, a major decrease of 68.

```
ARMED_FORCES_SERV_DATES        DATES_OF_MILITARY_SRVICE
                           in  UNDERGRAD_APPLICATION;

ATTEN_INSTIT_DATES             ATTEN_COLLEGE_FROM_DATE,
                               ATTEN_COLLEGE_TO_DATE
                           in  UNDERGRAD_APPLICATION;

ATTEN_INSTIT_NAME              ATTEN_COLLEGE_NAME
                           in  UNDERGRAD_APPLICATION;

COLLEGE_GRAD_SCHOOL            YOUR_COLLEGE
                           in  A_PASS_F_GRADING_OPTION;
                               SCH
                           in  CLASS_BOOK;

COUNTRY_OF_CITIZENSHIP         US_CITIZEN
                           in  UNDERGRAD_APPLICATION;

COURSE_TITLE                   COURSE_NAME
                           in  CLASS_BOOK,
                               GRADE_CHANGE;

DATE_ENROLLED                  SEMESTER_OF_ADMISSION,
                               YEAR_OF_ADMISSION
                           in  GRAD_APPLICATION;
                               ENTER_SEMESTER,
                               ENTER_YEAR
                           in  UNDERGRAD_APPLICATION;

DATE_OF_APPLICATION            APPLICATION_DATE
                           in  GRAD_APPLICATION;

DATE_OF_KANSAS_RESIDENCY       DATE_OF_KANSAS_RESIDENT
                           in  GRAD_APPLICATION;

EFFECTIVE_DATE                 TODAYS_DATE
                           in  A_PASS_F_GRADING_OPTION;

GRADE                          NEW_GRADE
                           in  GRADE_CHANGE;
                               FINAL
                           in  CLASS_BOOK;

MEETING_DAYS                   DAYS
                           in  SCHEDULE_OF_CLASSES;
MEETING_PLACE                  ROOM_NUMBER
                           in  SCHEDULE_OF_CLASSES;
MEETING_TIME                   FROM_HOUR,
                               TO_HOUR
                           in  SCHEDULE_OF_CLASSES;

OTHER_NAME                     OTHER_LAST_NAME
                           in  UNDERGRAD_APPLICATION;

PLACE_OF_BIRTH_CITY            PLACE_OF_BIRTH
                           in  GRAD_APPLICATION;

PLACE_OF_BIRTH_STATE           PLACE_OF_BIRTH
```

```
                        in GRAD_APPLICATION;

RELATIVE_ADDRESS_CITY      PERSON_IN_EMER_CITY
                        in GRAD_APPLICATION;
                           PARENT_CITY
                        in STUDENT_FILE;

RELATIVE_ADDRESS_STATE     PERSON_IN_EMER_COUNTY,
                           PERSON_IN_EMER_STATE
                        in GRAD_APPLICATION;
                           PARENT_STATE
                        in STUDENT_FILE;
RELATIVE_ADDRESS_STREET    PERSON_IN_EMER_STREET
                        in GRAD_APPLICATION;
                           PARENT_STREET
                        in STUDENT_FILE;

RELATIVE_ADDRESS_ZIP       PARENT_ZIP
                        in STUDENT_FILE;

RELATIVE_NAME              PERSON_IN_EMERGENCY_LAST_NA,
                           PERSON_IN_EMERGENCY_FIRST_NA,
                           PERSON_IN_EMERGENCY_MID_NA
                        in GRAD_APPLICATION;
                           PARENT_NAME
                        in STUDENT_FILE;

RELATIVE_TELEPHONE         PARENT_PHONE
                        in STUDENT_FILE;

SOCIAL_SECURITY_NUMBER     SOCIAL_SECURITY
                        in UNDERGRAD_APPLICATION;

STATUS_COLLEGE             STUDENT_SCHOOL
                        in TELEPHONE_DIRECTORY;

STATUS_DEGREE_SOUGHT       DEGREE_SELECTION
                        in UNDERGRAD_APPLICATION;
                           DEGREE
                        in GRAD_APPLICATION;

STATUS_LEVEL               S_CL
                        in CLASS_BOOK;
                           STUDENT_CLASS_ID
                        in TELEPHONE_DIRECTORY;

STUDENT_ADDRESS_CITY       LOCAL_ADDRESS_CITY
                        in STUDENT_FILE;
                           ADDRESS_CITY
                        in VEHICLE_REG_FORM;
                           PRESENT_ADDRESS_CITY
                        in GRAD_APPLICATION;

STUDENT_ADDRESS_STATE      LOCAL_ADDRESS_STATE
                        in STUDENT_FILE;
                           ADDRESS_STATE
                        in VEHICLE_REG_FORM;
                           PRESENT_ADDRESS_STATE
```

```
                        in GRAD_APPLICATION;

STUDENT_ADDRESS_STREET     LOCAL_ADDRESS_STREET
                        in STUDENT_FILE;
                           ADDRESS_STREET
                        in VEHICLE_REGISTRATION;
                           PRESENT_ADDRESS_STREET
                        in GRAD_APPLICATION;

STUDENT_ADDRESS_ZIP        LOCAL_ADDRESS_ZIP
                        in STUDENT_FILE;
                           ADDRESS_ZIP
                        in VEHICLE_REG_FORM;
                           PRESENT_ADDRESS_ZIP
                        in GRAD_APPLICATION;

STUDENT_NAME               NAME
                        in ASSIGNMENT_ENROLLMENT,
                           A_PASS_F_GRADING_OPTION,
                           CLASS_ENROLLMENT,
                           CHANGE_OF_ENROLLMENT,
                           FOREIGN_STUDENT_FINANCES;
                           NAME_LAST,
                           NAME_FIRST,
                           NAME_MIDDLE
                        in UNDERGRAD_APPLICATION,
                           GRAD_APPLICATION;

STUDENT_PERM_AD_CITY       PERMANENT_ADDRESS_CITY
                        in GRAD_APPLICATION;
                           HOME_ADDRESS_CITY
                        in UNDERGRAD_APPLICATION;
                           ADDRESS
                        in FOREIGN_STUDENT_FINANCES;

STUDENT_PERM_AD_COUNTY     PERMANENT_ADDRESS_COUNTY
                        in GRAD_APPLICATION;
                           HOME_ADDRESS_COUNTY
                        in UNDERGRAD_APPLICATION;
                           ADDRESS
                        in FOREIGN_STUDENT_FINANCES;

STUDENT_PERM_AD_STATE      PERMANENT_ADDRESS_STATE
                        in GRAD_APPLICATION;
                           HOME_ADDRESS_STATE
                        in UNDERGRAD_APPLICATION;
                           ADDRESS
                        in FOREIGN_STUDENT_FINANCES;

STUDENT_PERM_AD_STREET     PERMANENT_ADDRESS_STREET
                        in GRAD_APPLICATION;
                           HOME_ADDRESS_STREET
                        in UNDERGRAD_APPLICATION;
                           ADDRESS
                        in FOREIGN_STUDENT_FINANCES;

STUDENT_PERM_AD_ZIP        PERMANENT_ADDRESS_ZIP
                        in GRAD_APPLICATION;
```

	HOME_ADDRESS_ZIP in UNDERGRAD_APPLICATION; ADDRESS in FOREIGN_STUDENT_FINANCES;
STUDENT_PHONE	PHONE_NUMBER in GRAD_APPLICATION; TELEPHONE in STUDENT_FILE, UNDERGRAD_APPLICATION;

Table 4.1 Synonym columns

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP1

DOCUMENT / COLUMN LISTING

ASSIGNMENT_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE
LINE_NUMBER
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
CREDIT_STATUS
ADVISORS_SIGNATURE
DEANS_SIGNATURE
SOCIAL_SECURITY_NUMBER
SEMESTER
YEAR
STUDENT_NAME

A_PASS_F_GRADING_OPTION

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

LINE_NUMBER
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
DEANS_SIGNATURE
ADVISORS_SIGNATURE
STUDENT_SIGNATURE
SOCIAL_SECURITY_NUMBER
SEMESTER
YEAR
DEL_SEL_A_PASS_F_OPTION
STUDENT_NAME
COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE

CLASS_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

SOCIAL_SECURITY_NUMBER
COLLEGE_CURRICULUM
ADVISOR
CREDIT_STATUS
CREDIT_HOURS
COURSE_NUMBER
STUDENT_NAME
EFFECTIVE_DATE

CHANGE_OF_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

SOCIAL_SECURITY_NUMBER
COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE

LINE_NUMBER
DROP_ADD_CHANGE
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
UND_GRAD_CREDIT
ADVISORS_SIGNATURE
DEANS_OFFICE
STUDENT_SIGNATURE
SEMESTER
YEAR
WITHDRAW_FROM_UNIVERSITY
STUDENT_NAME

CLASS_BOOK

DOCUMENT ATTRIBUTES :
LOCATION . OUTPUT

LINE_NUMBER
COURSE_NUMBER
SEMESTER
MEETING_PLACE
SOCIAL_SECURITY_NUMBER
STUDENT_NAME
CREDIT_STATUS
CREDIT_HOURS
MEETING_TIME
MEETING_DAYS
COURSE_TITLE
COLLEGE_GRAD_SCHOOL
STATUS_LEVEL

CLASS_GRADES

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

COURSE_NUMBER
SOCIAL_SECURITY_NUMBER
GRADE
INSTRUCTORS_SIGNATURE

FOREIGN_STUDENT_FINANCES

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

STUDENT_PERM_AD_STREET
STUDENT_PERM_AD_CITY
STUDENT_PERM_AD_COUNTY
STUDENT_PERM_AD_STATE
STUDENT_PERM_AD_ZIP
SIGNATURE
DATE
STUDENT_NAME

GRADE_CHANGE

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

STUDENT_NAME

SOCIAL_SECURITY_NUMBER
PREV_GRADE
CREDIT_HOURS
COURSE_NUMBER
SEMESTER
YEAR
DATE
INSTRUCTOR
COURSE_TITLE
GRADE

GRAD_APPLICATION

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

MAJOR
SOCIAL_SECURITY_NUMBER
SEX
OTHER_NAMES
DATE_OF_BIRTH
COUNTRY_OF_CITIZENSHIP
VISA_TYPE
ATTEN_INSTIT_NAME
ATTEN_INSTIT_LOCATION
ATTEN_INSTIT_DATES
ATTEN_INSTIT_DIPLOMA
ATTEN_INSTIT_YEAR_RECEIVED
POSITION_HELD_NATURE
POSITION_HELD_COMPANY
POSITION_HELD_LOCATION
POSITION_HELD_DATES
ARMED_FORCES_SERV_DATES
REF_INSTR_NAME
REF_INSTR_POSITION
REF_INSTR_ADDRESS
WAIVE_RIGHT_OF_ACCESS
SCHOL_HONOR_PRIZES
ETHNIC_RACIAL_STATUS
APPLICANTS_SIGNATURE
SPEC_STUD_SIGNATURE
SPEC_STUD_APPLICATION_DATE
ACCEPTED_TO_DEGREE
TYPE_OF_ADMISSION
FOREIGN_ST_SUPPORT_SCHOLAR
FOREIGN_ST_SUPPORT_LOAN
FOREIGN_ST_SUPPORT_PERSONAL
FOREIGN_ST_SUPPORT_SUMMER_EX
TOTAL_IN_STATE_FEES
TOTAL_OUT_STATE_FEES
DATE_OF_APPLICATION
STUDENT_NAME
STUDENT_PHONE
PLACE_OF_BIRTH_CITY
PLACE_OF_BIRTH_STATE
RELATIVE_NAME
RELATIVE_ADDRESS_STREET

```
                    RELATIVE_ADDRESS
                    RELATIVE_ADDRESS_STATE
                    DATE_OF_KANSAS_RESIDENCY
                    STUDENT_ADDRESS_STREET
                    STUDENT_ADDRESS_CITY
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP
                    STUDENT_PERM_AD_STREET
                    STUDENT_PERM_AD_CITY
                    STUDENT_PERM_AD_COUNTY
                    STUDENT_PERM_AD_STATE
                    STUDENT_PERM_AD_ZIP
                    STATUS_DEGREE_SOUGHT
                    DATE_ENROLLED

SCHEDULE_OF_CLASSES
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                    LINE_NUMBER
                    COURSE_NUMBER
                    COURSE_TITLE
                    CREDIT_NO_CREDIT
                    CREDIT_HOURS
                    INSTRUCTOR
                    MEETING_DAYS
                    MEETING_PLACE
                    MEETING_TIME

STUDENT_GRADE_REPORT
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . OUTPUT
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_NAME
                    STUDENT_ADDRESS_STREET
                    STUDENT_ADDRESS_CITY
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
                    CREDIT_HOURS
                    GRADE
                    TOTAL_CR_HOURS
                    RESIDENT_GPA
                    TRANSFER_GPA

STUDENT_GRADE_HISTORY
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                    SOCIAL_SECURITY_NUMBER
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
                    CREDIT_HOURS
                    GRADE
```

```
DOCUMENT  HANDLER  VER  0.0      RUN UNIVERSITY EXAMPLE STEP1
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

          STUDENT_FILE
                                        DOCUMENT  ATTRIBUTES  :
                                              LOCATION . RESIDENT
                              SOCIAL_SECURITY_NUMBER
                              STUDENT_NAME
                              SEX
                              STUDENT_PERM_AD_STREET
                              STUDENT_PERM_AD_CITY
                              STUDENT_PERM_AD_COUNTY
                              STUDENT_PERM_AD_STATE
                              STUDENT_PERM_AD_ZIP
                              PLACE_OF_BIRTH_CITY
                              PLACE_OF_BIRTH_STATE
                              DATE_OF_BIRTH
                              GRADUATED_NAME
                              GRADUATED_CITY
                              GRADUATED_STATE
                              GRADUATED_DATE
                              DATE_ENROLLED
                              DATE_PROJ_GRAD
                              STATUS_COLLEGE
                              STATUS_LEVEL
                              STATUS_PRIMARY_MAJOR
                              STATUS_SECONDARY_MAJOR
                              STATUS_ADVISOR
                              STATUS_DEGREE_SOUGHT
                              MARITAL_STATUS
                              VEHICLE_STATE
                              VEHICLE_LICENSE_NUMBER
                              VEHICLE_MAKE
                              VEHICLE_YEAR
                              VEHICLE_PERMIT_NUMBER
                              STUDENT_PHONE
                              RELATIVE_NAME
                              RELATIVE_ADDRESS_STREET
                              RELATIVE_ADDRESS_CITY
                              RELATIVE_ADDRESS_STATE
                              RELATIVE_ADDRESS_ZIP
                              RELATIVE_TELEPHONE
                              STUDENT_ADDRESS_STREET
                              STUDENT_ADDRESS_CITY
                              STUDENT_ADDRESS_STATE
                              STUDENT_ADDRESS_ZIP

          TELEPHONE_DIRECTORY
                                        DOCUMENT  ATTRIBUTES  :
                                              LOCATION . OUTPUT
                              STUDENT_NAME
                              STUDENT_ADDRESS_STREET
                              STUDENT_PHONE
                              STATUS_LEVEL
                              STATUS_COLLEGE
                              STUDENT_PERM_AD_STREET
                              STUDENT_PERM_AD_CITY
```

```
DOCUMENT  HANDLER  VER  0.0      RUN UNIVERSITY EXAMPLE STEP1
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                        STUDENT_PERM_AD_STATE

UNDERGRAD_APPLICATION
                                    DOCUMENT  ATTRIBUTES  :
                                          LOCATION . INPUT
                        OTHER_NAMES
                        DATE_OF_BIRTH
                        AGE
                        PLACE_OF_BIRTH_CITY
                        PLACE_OF_BIRTH_STATE
                        SEX
                        VISA_TYPE
                        RELATIVE_NAME
                        RELATIVE_TELEPHONE
                        RELATIVE_ADDRESS_STREET
                        RELATIVE_ADDRESS_CITY
                        RELATIVE_ADDRESS_STATE
                        RELATIVE_ADDRESS_ZIP
                        DATE_OF_KANSAS_RESIDENCY
                        DATE_OF_PARENTS_KS_RESIDENCY
                        DEPENDENT_OF_MILITARY
                        HIGH_SCHOOL_NAME
                        HIGH_SCHOOL_CITY
                        HIGH_SCHOOL_STATE
                        HIGH_SCHOOL_DATE_FROM
                        HIGH_SCHOOL_DATE_TO
                        NOW_IN_COLLEGE
                        EVER_BEEN_IN_COLLEGE
                        ACT_DATE
                        CURRICULUM
                        CURRICULUM_NUMBER
                        PLAN_TO_COMPLETE_IN_KSU
                        PREV_ATTEN_KSU
                        DATE_ATTEN_KSU
                        TYPE_OF_PREV_PROGRAM
                        RELATIVE_ATTEN_KSU
                        RELATIVE_ATTEN_KSU_DATES
                        ETHNIC_RACIAL_STATUS
                        RELIGIOUS_PREFERENCE
                        APPLICANTS_SIGNATURE
                        DATE_OF_APPLICATION
                        STUDENT_NAME
                        STUDENT_PHONE
                        COUNTRY_OF_CITIZENSHIP
                        STUDENT_PERM_AD_STREET
                        STUDENT_PERM_AD_CITY
                        STUDENT_PERM_AD_COUNTY
                        STUDENT_PERM_AD_STATE
                        STUDENT_PERM_AD_ZIP
                        SOCIAL_SECURITY_NUMBER
                        STATUS_DEGREE_SOUGHT
                        ARMED_FORCES_SERV_DATES
                        ATTEN_INSTIT_NAME
                        ATTEN_INSTIT_DATES
                        DATE_ENROLLED
```

```
DOCUMENT  HANDLER  VER  0.0      RUN UNIVERSITY EXAMPLE STEP1
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

        VEHICLE_REG_FORM
                                        DOCUMENT  ATTRIBUTES  :
                                                LOCATION . INPUT
                        VEHICLE_STATE
                        VEHICLE_LICENSE_NUMBER
                        VEHICLE_MAKE
                        VEHICLE_YEAR
                        VEHICLE_PERMIT_NUMBER
                        SOCIAL_SECURITY_NUMBER
                        STUDENT_ADDRESS_STREET
                        STUDENT_ADDRESS_CITY
                        STUDENT_ADDRESS_STATE
                        STUDENT_ADDRESS_ZIP
```

figure 4.5

Deletion of Insignificant Columns

While analyzing the columns one by one, the columns in table 4.2 were declared insignificant for the UARS with respect to inclusion in the data-base. However, this does not mean that these columns are redundant in the original documents.

Figure 4.6 is a complete list of all the documents in the system after the insignificant columns are deleted. The total number of columns is now 260 and there are 117 distinct columns, a decrease of 12 columns when compared to the last step.

COLUMN	IN DOCUMENT
ADVISORS_SIGNATURE	ASSIGNMENT_ENROLLMENT, A_PASS_F_GRADING_OPTION, CHANGE_OF_ENROLLMENT
APPLICANTS_SIGNATURE	GRAD_APPLICATION, UNDERGRAD_APPLICATION
DATE	GRADE_CHANGE
DEANS_OFFICE	CHANGE_OF_ENROLLMENT
DEANS_SIGNATURE	ASSIGNMENT_ENROLLMENT, A_PASS_F_GRADING_OPTION
INSTRUCTOR	GRADE_CHANGE
INSTRUCTORS_SIGNATURE	CLASS_GRADES
REF_INSTR_ADDRESS	GRAD_APPLICATION
REF_INSTR_NAME	GRAD_APPLICATION
REF_INSTR_POSITION	GRAD_APPLICATION
SIGNATURE	FOREIGN_STUDENT_FINANCES
SPEC_STUD_APPLICATION_DATE	GRAD_APPLICATION
SPEC_STUD_SIGNATURE	GRAD_APPLICATION
STUDENT_SIGNATURE	CHANGE_OF_ENROLLMENT, A_PASS_F_GRADING_OPTION

table 4.2 : insignificant columns

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP2

DOCUMENT / COLUMN LISTING

ASSIGNMENT_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE
LINE_NUMBER
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
CREDIT_STATUS
SOCIAL_SECURITY_NUMBER
SEMESTER
YEAR
STUDENT_NAME

A_PASS_F_GRADING_OPTION

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

LINE_NUMBER
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
SOCIAL_SECURITY_NUMBER
SEMESTER
YEAR
DEL_SEL_A_PASS_F_OPTION
STUDENT_NAME
COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE

CLASS_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

SOCIAL_SECURITY_NUMBER
COLLEGE_CURRICULUM
ADVISOR
CREDIT_STATUS
CREDIT_HOURS
COURSE_NUMBER
STUDENT_NAME
EFFECTIVE_DATE

CHANGE_OF_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

SOCIAL_SECURITY_NUMBER
COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE
LINE_NUMBER
DROP_ADD_CHANGE
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS

```
                    UND_GRAD_CREDIT
                    SEMESTER
                    YEAR
                    WITHDRAW_FROM_UNIVERSITY
                    STUDENT_NAME

          CLASS_BOOK
                                DOCUMENT ATTRIBUTES :
                                      LOCATION . OUTPUT
                    LINE_NUMBER
                    COURSE_NUMBER
                    SEMESTER
                    MEETING_PLACE
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_NAME
                    CREDIT_STATUS
                    CREDIT_HOURS
                    MEETING_TIME
                    MEETING_DAYS
                    COURSE_TITLE
                    COLLEGE_GRAD_SCHOOL
                    STATUS_LEVEL

          CLASS_GRADES
                                DOCUMENT ATTRIBUTES :
                                      LOCATION . INPUT
                    COURSE_NUMBER
                    SOCIAL_SECURITY_NUMBER
                    GRADE

          FOREIGN_STUDENT_FINANCES
                                DOCUMENT ATTRIBUTES :
                                      LOCATION . INPUT
                    STUDENT_PERM_AD_STREET
                    STUDENT_PERM_AD_CITY
                    STUDENT_PERM_AD_COUNTY
                    STUDENT_PERM_AD_STATE
                    STUDENT_PERM_AD_ZIP
                    DATE
                    STUDENT_NAME

          GRADE_CHANGE
                                DOCUMENT ATTRIBUTES :
                                      LOCATION . INPUT
                    STUDENT_NAME
                    SOCIAL_SECURITY_NUMBER
                    PREV_GRADE
                    CREDIT_HOURS
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
                    COURSE_TITLE
                    GRADE

          GRAD_APPLICATION
```

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

MAJOR
SOCIAL_SECURITY_NUMBER
SEX
OTHER_NAMES
DATE_OF_BIRTH
COUNTRY_OF_CITIZENSHIP
VISA_TYPE
ATTEN_INSTIT_NAME
ATTEN_INSTIT_LOCATION
ATTEN_INSTIT_DATES
ATTEN_INSTIT_DIPLOMA
ATTEN_INSTIT_YEAR_RECEIVED
POSITION_HELD_NATURE
POSITION_HELD_COMPANY
POSITION_HELD_LOCATION
POSITION_HELD_DATES
ARMED_FORCES_SERV_DATES
WAIVE_RIGHT_OF_ACCESS
SCHOL_HONOR_PRIZES
ETHNIC_RACIAL_STATUS
ACCEPTED_TO_DEGREE
TYPE_OF_ADMISSION
FOREIGN_ST_SUPPORT_SCHOLAR
FOREIGN_ST_SUPPORT_LOAN
FOREIGN_ST_SUPPORT_PERSONAL
FOREIGN_ST_SUPPORT_SUMMER_EX
TOTAL_IN_STATE_FEES
TOTAL_OUT_STATE_FEES
DATE_OF_APPLICATION
STUDENT_NAME
STUDENT_PHONE
PLACE_OF_BIRTH_CITY
PLACE_OF_BIRTH_STATE
RELATIVE_NAME
RELATIVE_ADDRESS_STREET
RELATIVE_ADDRESS
RELATIVE_ADDRESS_STATE
DATE_OF_KANSAS_RESIDENCY
STUDENT_ADDRESS_STREET
STUDENT_ADDRESS_CITY
STUDENT_ADDRESS_STATE
STUDENT_ADDRESS_ZIP
STUDENT_PERM_AD_STREET
STUDENT_PERM_AD_CITY
STUDENT_PERM_AD_COUNTY
STUDENT_PERM_AD_STATE
STUDENT_PERM_AD_ZIP
STATUS_DEGREE_SOUGHT
DATE_ENROLLED

SCHEDULE_OF_CLASSES

DOCUMENT ATTRIBUTES :
LOCATION . RESIDENT

```
                    LINE_NUMBER
                    COURSE_NUMBER
                    COURSE_TITLE
                    CREDIT_NO_CREDIT
                    CREDIT_HOURS
                    INSTRUCTOR
                    MEETING_DAYS
                    MEETING_PLACE
                    MEETING_TIME

STUDENT_GRADE_REPORT
                              DOCUMENT ATTRIBUTES :
                                    LOCATION . OUTPUT
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_NAME
                    STUDENT_ADDRESS_STREET
                    STUDENT_ADDRESS_CITY
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
                    CREDIT_HOURS
                    GRADE
                    TOTAL_CR_HOURS
                    RESIDENT_GPA
                    TRANSFER_GPA

STUDENT_GRADE_HISTORY
                              DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                    SOCIAL_SECURITY_NUMBER
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
                    CREDIT_HOURS
                    GRADE

STUDENT_FILE
                              DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_NAME
                    SEX
                    STUDENT_PERM_AD_STREET
                    STUDENT_PERM_AD_CITY
                    STUDENT_PERM_AD_COUNTY
                    STUDENT_PERM_AD_STATE
                    STUDENT_PERM_AD_ZIP
                    PLACE_OF_BIRTH_CITY
                    PLACE_OF_BIRTH_STATE
                    DATE_OF_BIRTH
                    GRADUATED_NAME
                    GRADUATED_CITY
                    GRADUATED_STATE
```

GRADUATED_DATE
DATE_ENROLLED
DATE_PROJ_GRAD
STATUS_COLLEGE
STATUS_LEVEL
STATUS_PRIMARY_MAJOR
STATUS_SECONDARY_MAJOR
STATUS_ADVISOR
STATUS_DEGREE_SOUGHT
MARITAL_STATUS
VEHICLE_STATE
VEHICLE_LICENSE_NUMBER
VEHICLE_MAKE
VEHICLE_YEAR
VEHICLE_PERMIT_NUMBER
STUDENT_PHONE
RELATIVE_NAME
RELATIVE_ADDRESS_STREET
RELATIVE_ADDRESS_CITY
RELATIVE_ADDRESS_STATE
RELATIVE_ADDRESS_ZIP
RELATIVE_TELEPHONE
STUDENT_ADDRESS_STREET
STUDENT_ADDRESS_CITY
STUDENT_ADDRESS_STATE
STUDENT_ADDRESS_ZIP

TELEPHONE_DIRECTORY

DOCUMENT ATTRIBUTES :
LOCATION . OUTPUT

STUDENT_NAME
STUDENT_ADDRESS_STREET
STUDENT_PHONE
STATUS_LEVEL
STATUS_COLLEGE
STUDENT_PERM_AD_STREET
STUDENT_PERM_AD_CITY
STUDENT_PERM_AD_STATE

UNDERGRAD_APPLICATION

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

OTHER_NAMES
DATE_OF_BIRTH
AGE
PLACE_OF_BIRTH_CITY
PLACE_OF_BIRTH_STATE
SEX
VISA_TYPE
RELATIVE_NAME
RELATIVE_TELEPHONE
RELATIVE_ADDRESS_STREET
RELATIVE_ADDRESS_CITY
RELATIVE_ADDRESS_STATE
RELATIVE_ADDRESS_ZIP

```
                    DATE_OF_KANSAS_RESIDENCY
                    DATE_OF_PARENTS_KS_RESIDENCY
                    DEPENDENT_OF_MILITARY
                    HIGH_SCHOOL_NAME
                    HIGH_SCHOOL_CITY
                    HIGH_SCHOOL_STATE
                    HIGH_SCHOOL_DATE_FROM
                    HIGH_SCHOOL_DATE_TO
                    NOW_IN_COLLEGE
                    EVER_BEEN_IN_COLLEGE
                    ACT_DATE
                    CURRICULUM
                    CURRICULUM_NUMBER
                    PLAN_TO_COMPLETE_IN_KSU
                    PREV_ATTEN_KSU
                    DATE_ATTEN_KSU
                    TYPE_OF_PREV_PROGRAM
                    RELATIVE_ATTEN_KSU
                    RELATIVE_ATTEN_KSU_DATES
                    ETHNIC_RACIAL_STATUS
                    RELIGIOUS_PREFERENCE
                    DATE_OF_APPLICATION
                    STUDENT_NAME
                    STUDENT_PHONE
                    COUNTRY_OF_CITIZENSHIP
                    STUDENT_PERM_AD_STREET
                    STUDENT_PERM_AD_CITY
                    STUDENT_PERM_AD_COUNTY
                    STUDENT_PERM_AD_STATE
                    STUDENT_PERM_AD_ZIP
                    SOCIAL_SECURITY_NUMBER
                    STATUS_DEGREE_SOUGHT
                    ARMED_FORCES_SERV_DATES
                    ATTEN_INSTIT_NAME
                    ATTEN_INSTIT_DATES
                    DATE_ENROLLED

VEHICLE_REG_FORM
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . INPUT
                    VEHICLE_STATE
                    VEHICLE_LICENSE_NUMBER
                    VEHICLE_MAKE
                    VEHICLE_YEAR
                    VEHICLE_PERMIT_NUMBER
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_ADDRESS_STREET
                    STUDENT_ADDRESS_CITY
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP
```

figure 4.6

Solving Undeclared Output Columns

Figure 4.7 lists all the columns that appear in output documents but do not appear in any input or resident document.

To solve this problem, the document CALC_GPA was added with the location of resident. Figure 4.8 is a partial list of the columns in the new document. At this point there are 17 documents in the system, with 117 distinct columns and a total of 269 columns.

```
DOCUMENT  HANDLER  VER  0.0      RUN  UNIVERSITY EXAMPLE STEP3
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                    SELECT      LISTING
               WITHOUT       LOCATION . INPUT
               WITHOUT       LOCATION . RESIDENT

   RESIDENT_GPA
   TOTAL_CR_HOURS
   TRANSFER_GPA
```

figure 4.7 : Undeclared Output Columns

```
DOCUMENT  HANDLER  VER  0.0      RUN  UNIVERSITY EXAMPLE STEP3
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-

              DOCUMENT / COLUMN  LISTING

CALC_GPA
                          DOCUMENT ATTRIBUTES  :
                                LOCATION . RESIDENT
                  SOCIAL_SECURITY_NUMBER
                  COURSE_NUMBER
                  SEMESTER
                  YEAR
                  CREDIT_HOURS
                  GRADE
                  RESIDENT_GPA
                  TRANSFER_GPA
                  TOTAL_CR_HOURS
```

figure 4.8 : The Document CALC_GPA

Specifying Keys

When no more synonyms are found, when all the insignificant columns are deleted and there are no undeclared output columns, it is time to specify the key columns in every document. Figure 4.9 is a complete listing of all the documents with the key columns indicated by an asterisk. In the case of the document VEHICLE_REG_FORM, there are three candidate keys. A vehicle is found according to its license number, its permit number or its student owner. Thus, two more copies of that document are added with the specification of the different keys.

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP4

DOCUMENT / COLUMN LISTING

* ASSIGNMENT_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE
LINE_NUMBER
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
CREDIT_STATUS
* SOCIAL_SECURITY_NUMBER
SEMESTER
YEAR
STUDENT_NAME

* A_PASS_F_GRADING_OPTION

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

LINE_NUMBER
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS
* SOCIAL_SECURITY_NUMBER
SEMESTER
YEAR
DEL_SEL_A_PASS_F_OPTION
STUDENT_NAME
COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE

* CLASS_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

* SOCIAL_SECURITY_NUMBER
COLLEGE_CURRICULUM
ADVISOR
CREDIT_STATUS
CREDIT_HOURS
COURSE_NUMBER
STUDENT_NAME
EFFECTIVE_DATE

* CHANGE_OF_ENROLLMENT

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

* SOCIAL_SECURITY_NUMBER
COLLEGE_GRAD_SCHOOL
EFFECTIVE_DATE
LINE_NUMBER
DROP_ADD_CHANGE
COURSE_NUMBER
COURSE_TITLE
CREDIT_HOURS

```
                    UND_GRAD_CREDIT
                    SEMESTER
                    YEAR
                    WITHDRAW_FROM_UNIVERSITY
                    STUDENT_NAME

  * CLASS_BOOK
                                DOCUMENT ATTRIBUTES :
                                        LOCATION . OUTPUT
                    LINE_NUMBER
                  * COURSE_NUMBER
                    SEMESTER
                    MEETING_PLACE
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_NAME
                    CREDIT_STATUS
                    CREDIT_HOURS
                    MEETING_TIME
                    MEETING_DAYS
                    COURSE_TITLE
                    COLLEGE_GRAD_SCHOOL
                    STATUS_LEVEL

  * CLASS_GRADES
                                DOCUMENT ATTRIBUTES :
                                        LOCATION . INPUT
                  * COURSE_NUMBER
                  * SOCIAL_SECURITY_NUMBER
                    GRADE

  * CALC_GPA
                                DOCUMENT ATTRIBUTES :
                                        LOCATION . RESIDENT
                  * SOCIAL_SECURITY_NUMBER
                    COURSE_NUMBER
                    SEMESTER
                    YEAR
                    CREDIT_HOURS
                    GRADE
                    RESIDENT_GPA
                    TRANSFER_GPA
                    TOTAL_CR_HOURS

  * FOREIGN_STUDENT_FINANCES
                                DOCUMENT ATTRIBUTES :
                                        LOCATION . INPUT
                    STUDENT_PERM_AD_STREET
                    STUDENT_PERM_AD_CITY
                    STUDENT_PERM_AD_COUNTY
                    STUDENT_PERM_AD_STATE
                    STUDENT_PERM_AD_ZIP
                    DATE
                  * STUDENT_NAME

  * GRADE_CHANGE
```

```
DOCUMENT  HANDLER  VER  0.0      RUN  UNIVERSITY EXAMPLE STEP4
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                                        DOCUMENT  ATTRIBUTES  :
                                                LOCATION . INPUT
                          STUDENT_NAME
                        * SOCIAL_SECURITY_NUMBER
                          PREV_GRADE
                          CREDIT_HOURS
                        * COURSE_NUMBER
                          SEMESTER
                          YEAR
                          COURSE_TITLE
                          GRADE

      * GRAD_APPLICATION
                                        DOCUMENT  ATTRIBUTES  :
                                                LOCATION . INPUT
                          MAJOR
                        * SOCIAL_SECURITY_NUMBER
                          SEX
                          OTHER_NAMES
                          DATE_OF_BIRTH
                          COUNTRY_OF_CITIZENSHIP
                          VISA_TYPE
                          ATTEN_INSTIT_NAME
                          ATTEN_INSTIT_LOCATION
                          ATTEN_INSTIT_DATES
                          ATTEN_INSTIT_DIPLOMA
                          ATTEN_INSTIT_YEAR_RECEIVED
                          POSITION_HELD_NATURE
                          POSITION_HELD_COMPANY
                          POSITION_HELD_LOCATION
                          POSITION_HELD_DATES
                          ARMED_FORCES_SERV_DATES
                          WAIVE_RIGHT_OF_ACCESS
                          SCHOL_HONOR_PRIZES
                          ETHNIC_RACIAL_STATUS
                          ACCEPTED_TO_DEGREE
                          TYPE_OF_ADMISSION
                          FOREIGN_ST_SUPPORT_SCHOLAR
                          FOREIGN_ST_SUPPORT_LOAN
                          FOREIGN_ST_SUPPORT_PERSONAL
                          FOREIGN_ST_SUPPORT_SUMMER_EX
                          TOTAL_IN_STATE_FEES
                          TOTAL_OUT_STATE_FEES
                          DATE_OF_APPLICATION
                          STUDENT_NAME
                          STUDENT_PHONE
                          PLACE_OF_BIRTH_CITY
                          PLACE_OF_BIRTH_STATE
                          RELATIVE_NAME
                          RELATIVE_ADDRESS_STREET
                          RELATIVE_ADDRESS
                          RELATIVE_ADDRESS_STATE
                          DATE_OF_KANSAS_RESIDENCY
                          STUDENT_ADDRESS_STREET
                          STUDENT_ADDRESS_CITY
```

```
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP
                    STUDENT_PERM_AD_STREET
                    STUDENT_PERM_AD_CITY
                    STUDENT_PERM_AD_COUNTY
                    STUDENT_PERM_AD_STATE
                    STUDENT_PERM_AD_ZIP
                    STATUS_DEGREE_SOUGHT
                    DATE_ENROLLED

* SCHEDULE_OF_CLASSES
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                * LINE_NUMBER
                  COURSE_NUMBER
                  COURSE_TITLE
                  CREDIT_NO_CREDIT
                  CREDIT_HOURS
                  INSTRUCTOR
                  MEETING_DAYS
                  MEETING_PLACE
                  MEETING_TIME

* STUDENT_GRADE_REPORT
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . OUTPUT
                * SOCIAL_SECURITY_NUMBER
                  STUDENT_NAME
                  STUDENT_ADDRESS_STREET
                  STUDENT_ADDRESS_CITY
                  STUDENT_ADDRESS_STATE
                  STUDENT_ADDRESS_ZIP
                  COURSE_NUMBER
                  SEMESTER
                  YEAR
                  CREDIT_HOURS
                  GRADE
                  TOTAL_CR_HOURS
                  RESIDENT_GPA
                  TRANSFER_GPA

* STUDENT_GRADE_HISTORY
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                * SOCIAL_SECURITY_NUMBER
                  COURSE_NUMBER
                  SEMESTER
                  YEAR
                  CREDIT_HOURS
                  GRADE

* STUDENT_FILE
                                DOCUMENT ATTRIBUTES :
                                    LOCATION . RESIDENT
                * SOCIAL_SECURITY_NUMBER
```

STUDENT_NAME
SEX
STUDENT_PERM_AD_STREET
STUDENT_PERM_AD_CITY
STUDENT_PERM_AD_COUNTY
STUDENT_PERM_AD_STATE
STUDENT_PERM_AD_ZIP
PLACE_OF_BIRTH_CITY
PLACE_OF_BIRTH_STATE
DATE_OF_BIRTH
GRADUATED_NAME
GRADUATED_CITY
GRADUATED_STATE
GRADUATED_DATE
DATE_ENROLLED
DATE_PROJ_GRAD
STATUS_COLLEGE
STATUS_LEVEL
STATUS_PRIMARY_MAJOR
STATUS_SECONDARY_MAJOR
STATUS_ADVISOR
STATUS_DEGREE_SOUGHT
MARITAL_STATUS
VEHICLE_STATE
VEHICLE_LICENSE_NUMBER
VEHICLE_MAKE
VEHICLE_YEAR
VEHICLE_PERMIT_NUMBER
STUDENT_PHONE
RELATIVE_NAME
RELATIVE_ADDRESS_STREET
RELATIVE_ADDRESS_CITY
RELATIVE_ADDRESS_STATE
RELATIVE_ADDRESS_ZIP
RELATIVE_TELEPHONE
STUDENT_ADDRESS_STREET
STUDENT_ADDRESS_CITY
STUDENT_ADDRESS_STATE
STUDENT_ADDRESS_ZIP

* TELEPHONE_DIRECTORY

DOCUMENT ATTRIBUTES :
LOCATION . OUTPUT

* STUDENT_NAME
* STUDENT_ADDRESS_STREET
STUDENT_PHONE
STATUS_LEVEL
STATUS_COLLEGE
STUDENT_PERM_AD_STREET
* STUDENT_PERM_AD_CITY
STUDENT_PERM_AD_STATE

* UNDERGRAD_APPLICATION

DOCUMENT ATTRIBUTES :
LOCATION . INPUT

```
                        OTHER_NAMES
                        DATE_OF_BIRTH
                        AGE
                        PLACE_OF_BIRTH_CITY
                        PLACE_OF_BIRTH_STATE
                        SEX
                        VISA_TYPE
                        RELATIVE_NAME
                        RELATIVE_TELEPHONE
                        RELATIVE_ADDRESS_STREET
                        RELATIVE_ADDRESS_CITY
                        RELATIVE_ADDRESS_STATE
                        RELATIVE_ADDRESS_ZIP
                        DATE_OF_KANSAS_RESIDENCY
                        DATE_OF_PARENTS_KS_RESIDENCY
                        DEPENDENT_OF_MILITARY
                        HIGH_SCHOOL_NAME
                        HIGH_SCHOOL_CITY
                        HIGH_SCHOOL_STATE
                        HIGH_SCHOOL_DATE_FROM
                        HIGH_SCHOOL_DATE_TO
                        NOW_IN_COLLEGE
                        EVER_BEEN_IN_COLLEGE
                        ACT_DATE
                        CURRICULUM
                        CURRICULUM_NUMBER
                        PLAN_TO_COMPLETE_IN_KSU
                        PREV_ATTEN_KSU
                        DATE_ATTEN_KSU
                        TYPE_OF_PREV_PROGRAM
                        RELATIVE_ATTEN_KSU
                        RELATIVE_ATTEN_KSU_DATES
                        ETHNIC_RACIAL_STATUS
                        RELIGIOUS_PREFERENCE
                        DATE_OF_APPLICATION
                        STUDENT_NAME
                        STUDENT_PHONE
                        COUNTRY_OF_CITIZENSHIP
                        STUDENT_PERM_AD_STREET
                        STUDENT_PERM_AD_CITY
                        STUDENT_PERM_AD_COUNTY
                        STUDENT_PERM_AD_STATE
                        STUDENT_PERM_AD_ZIP
                      * SOCIAL_SECURITY_NUMBER
                        STATUS_DEGREE_SOUGHT
                        ARMED_FORCES_SERV_DATES
                        ATTEN_INSTIT_NAME
                        ATTEN_INSTIT_DATES
                        DATE_ENROLLED

    * VEHICLE_REG_FORM
                                        DOCUMENT ATTRIBUTES :
                                              LOCATION . INPUT
                        VEHICLE_STATE
                        VEHICLE_LICENSE_NUMBER
```

```
                    VEHICLE_MAKE
                    VEHICLE_YEAR
                    VEHICLE_PERMIT_NUMBER
                  * SOCIAL_SECURITY_NUMBER
                    STUDENT_ADDRESS_STREET
                    STUDENT_ADDRESS_CITY
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP

* VEHICLE_REG_FORM_1
                                DOCUMENT ATTRIBUTES :
                                      LOCATION . INPUT
                  * VEHICLE_STATE
                  * VEHICLE_LICENSE_NUMBER
                    VEHICLE_MAKE
                    VEHICLE_YEAR
                    VEHICLE_PERMIT_NUMBER
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_ADDRESS_STREET
                    STUDENT_ADDRESS_CITY
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP

* VEHICLE_REG_FORM_2
                                DOCUMENT ATTRIBUTES :
                                      LOCATION . INPUT
                    VEHICLE_STATE
                    VEHICLE_LICENSE_NUMBER
                    VEHICLE_MAKE
                    VEHICLE_YEAR
                  * VEHICLE_PERMIT_NUMBER
                    SOCIAL_SECURITY_NUMBER
                    STUDENT_ADDRESS_STREET
                    STUDENT_ADDRESS_CITY
                    STUDENT_ADDRESS_STATE
                    STUDENT_ADDRESS_ZIP
```

figure 4.9 : The documents with specified keys

Deriving the Schema in Third Normal Form

Using the PREPARE command made available by the document handler, functional dependencies are formed by treating each document as a relation. The left hand side is derived from all the key columns specified for any document. The other non-key columns are used in the right hand side of the functional dependency.

However, because of restrictions imposed by the Bernstein's algorithm program currently implemented at Kansas State University, column names may not exceed three letters in length. To help overcome this restriction, the document handler assigns an abbreviated name to each column name and prints out a dictionary. Figure 4.10 is the dictionary printed for the columns used in the UARS. The functional dependencies are printed on a file in a format acceptable by Bernstein's algorithm program. Figure 4.11 shows the functional dependencies used in the UARS example. Figure 4.12 is the resulting third normal form schema after the execution of Bernstein's algorithm.

DOCUMENT HANDLER VER 0.0 RUN UNIVERSITY EXAMPLE STEP5

DICTIONARY - BERNSTEIN"S ALGORYTHM

ADVISOR	A00
ATTEN_INSTIT_NAME	A01
ATTEN_INSTIT_LOCATION	A02
ATTEN_INSTIT_DATES	A03
ATTEN_INSTIT_DIPLOMA	A04
ATTEN_INSTIT_YEAR_RECEIVED	A05
ARMED_FORCES_SERV_DATES	A06
ACCEPTED_TO_DEGREE	A07
AGE	A08
ACT_DATE	A09
COLLEGE_GRAD_SCHOOL	C00
COURSE_NUMBER	C01
COURSE_TITLE	C02
CREDIT_HOURS	C03
CREDIT_STATUS	C04
COLLEGE_CURRICULUM	C05
COUNTRY_OF_CITIZENSHIP	C06
CREDIT_NO_CREDIT	C07
CURRICULUM	C08
CURRICULUM_NUMBER	C09
DEL_SEL_A_PASS_F_OPTION	D00
DROP_ADD_CHANGE	D01
DATE	D02
DATE_OF_BIRTH	D03
DATE_OF_APPLICATION	D04
DATE_OF_KANSAS_RESIDENCY	D05
DATE_ENROLLED	D06
DATE_PROJ_GRAD	D07
DATE_OF_PARENTS_KS_RESIDENCY	D08
DEPENDENT_OF_MILITARY	D09
DATE_ATTEN_KSU	D10
EFFECTIVE_DATE	E00
ETHNIC_RACIAL_STATUS	E01
EVER_BEEN_IN_COLLEGE	E02
FOREIGN_ST_SUPPORT_SCHOLAR	F00
FOREIGN_ST_SUPPORT_LOAN	F01
FOREIGN_ST_SUPPORT_PERSONAL	F02
FOREIGN_ST_SUPPORT_SUMMER_EX	F03
GRADE	G00
GRADUATED_NAME	G01
GRADUATED_CITY	G02
GRADUATED_STATE	G03
GRADUATED_DATE	G04
HIGH_SCHOOL_NAME	H00
HIGH_SCHOOL_CITY	H01
HIGH_SCHOOL_STATE	H02
HIGH_SCHOOL_DATE_FROM	H03
HIGH_SCHOOL_DATE_TO	H04
INSTRUCTOR	I00
LINE_NUMBER	L00
MEETING_PLACE	M00

MEETING_TIME	M01
MEETING_DAYS	M02
MAJOR	M03
MARITAL_STATUS	M04
NOW_IN_COLLEGE	N00
OTHER_NAMES	O00
OTHER_NAMES	O01
PREV_GRADE	P00
POSITION_HELD_NATURE	P01
POSITION_HELD_COMPANY	P02
POSITION_HELD_LOCATION	P03
POSITION_HELD_DATES	P04
PLACE_OF_BIRTH_CITY	P05
PLACE_OF_BIRTH_STATE	P06
PLAN_TO_COMPLETE_IN_KSU	P07
PREV_ATTEN_KSU	P08
RELATIVE_NAME	R00
RELATIVE_ADDRESS_STREET	R01
RELATIVE_ADDRESS	R02
RELATIVE_ADDRESS_STATE	R03
RESIDENT_GPA	R04
RELATIVE_ADDRESS_CITY	R05
RELATIVE_ADDRESS_ZIP	R06
RELATIVE_TELEPHONE	R07
RELATIVE_ADDRESS_CITY	R08
RELATIVE_ADDRESS_STATE	R09
RELATIVE_ATTEN_KSU	R10
RELATIVE_ATTEN_KSU_DATES	R11
RELIGIOUS_PREFERENCE	R12
SOCIAL_SECURITY_NUMBER	S00
SEMESTER	S01
STUDENT_NAME	S02
STATUS_LEVEL	S03
STUDENT_PERM_AD_STREET	S04
STUDENT_PERM_AD_CITY	S05
STUDENT_PERM_AD_COUNTY	S06
STUDENT_PERM_AD_STATE	S07
STUDENT_PERM_AD_ZIP	S08
SEX	S09
SCHOL_HONOR_PRIZES	S10
STUDENT_PHONE	S11
STUDENT_ADDRESS_STREET	S12
STUDENT_ADDRESS_CITY	S13
STUDENT_ADDRESS_STATE	S14
STUDENT_ADDRESS_ZIP	S15
STATUS_DEGREE_SOUGHT	S16
STATUS_COLLEGE	S17
STATUS_PRIMARY_MAJOR	S18
STATUS_SECONDARY_MAJOR	S19
STATUS_ADVISOR	S20
TYPE_OF_ADMISSION	T00
TOTAL_IN_STATE_FEES	T01
TOTAL_OUT_STATE_FEES	T02
TOTAL_CR_HOURS	T03
TRANSFER_GPA	T04
TYPE_OF_PREV_PROGRAM	T05

UND_GRAD_CREDIT	U00
VISA_TYPE	V00
VEHICLE_STATE	V01
VEHICLE_LICENSE_NUMBER	V02
VEHICLE_MAKE	V03
VEHICLE_YEAR	V04
VEHICLE_PERMIT_NUMBER	V05
WITHDRAW_FROM_UNIVERSITY	W00
WAIVE_RIGHT_OF_ACCESS	W01
YEAR	Y00

figure 4.10 : Dictionary for Bernstein's Algorithm

```
S00>C00,E00,L00,C01,C02,C03,C04,S01,Y00,S02;
S00>L00,C01,C02,C03,S01,Y00,D00,S02,C00,E00;
S00>C05,A00,C04,C03,C01,S02,E00;
S00>C00,E00,L00,D01,C01,C02,C03,U00,S01,Y00,W00,S02;
C01>L00,S01,M00,S00,S02,C04,C03,M01,M02,C02,C00,S03;
C01,S00>G00;
S02>S04,S05,S06,S07,S08,D02;
S00,C01>S02,P00,C03,S01,Y00,C02,G00;
S00>M03,S09,O00,D03,C06,V00,A01,A02,A03,A04,A05,P01,
    P02,P03,P04,A06,W01,S10,E01,A07,T00,F00,F01,F02,
    F03,T01,T02,D04,S02,S11,P05,P06,R00,R01,R02,R03,
    D05,S12,S13,S14,S15,S04,S05,S06,S07,S08,S16,D06;
L00>C01,C02,C07,C03,I00,M02,M00,M01;
S00>S02,S12,S13,S14,S15,C01,S01,Y00,C03,G00,T03,R04,
    T04;
S00>C01,S01,Y00,C03,G00;
S00>S02,S09,S04,S05,S06,S07,S08,P05,P06,D03,G01,G02,
    G03,G04,D06,D07,S17,S03,S18,S19,S20,S16,M04,V01,
    V02,V03,V04,V05,S11,R00,R01,R05,R03,R06,R07,S12,
    S13,S14,S15;
S02,S12>S05,S11,S03,S17,S04,S07;
S00>O01,D03,A08,P05,P06,S09,V00,R00,R07,R01,R08,R09,
    R06,D05,D08,D09,H00,H01,H02,H03,H04,N00,E02,A09,
    C08,C09,P07,PO8,D10,T05,R10,R11,E01,R12,D04,S02,
    S11,C06,S04,S05,S06,S07,S08,S16,A06,A01,A03,D06;
S00>V01,V02,V03,V04,V05,S12,S13,S14,S15;
V01,V02>S00,V03,V04,V05,S12,S13,S14,S15;
V05>S00,V01,V02,V03,V04,S12,S13,S14,S15;
```

figure 4.11 Functional dependencies for the UARS

```
(V05),(S00),(L00),(C01),(V02 V01) > S00,V03,V04,S12,S13,S14,
       S15,L00,D00,C05,A00,E00,D01,U00,W00,M03,D00,A02,A04,
       A05,P01,P02,P03,P04,W01,S10,A07,T00,F00,F01,F02,F03,
       T01,T02,R02,T03,R04,T04,S01,Y00,C03,G00,G01,G02,G03,
       G04,D07,S18,S19,S20,M04,R05,R03,D01,D03,A08,P05,P06,
       S09,V00,R00,R07,R01,R08,R09,R06,D05,D08,D09,H00,H01,
       H02,H03,H04,N00,E02,A09,C08,C09,P07,P08,D10,T05,R10,
       R11,E01,R12,D04,S02,C06,S16,A06,A01,A03,D06,C01,C02,
       C07,I00,M02,M00,M01,V02,V01,C04,C00,P00,V05;
(S02) > S06,S08,D02,S05,S04,S07;
(S02 S12) > S17,S03,S11;
```

figure 4.12 : The Schema in Third Normal Form

Conclusion

Summary

This report described the methodology of logical data base design and detailed the implementation of a tool named Document Handler.

An extensive example demonstrated the design process and the use of the Document Handler.

The document handler was engineered as a very powerful tool necessary for the successful logical design of data bases. It was written in PASCAL to provide portability features and very strong data structures. Although the program consists of more than 3,000 lines of code, maintenance is very easy. New commands and features may be added without any change to the code of the currently existing features. The total storage requirement of the current version is 34704 bytes. This number may be greatly decreased if the TEACH command is taken out. Approximately 250 hours of time were used in the design and implementation of the program.

Enhancements

The Document Handler can be enhanced by adding several more commands and features such as:

1) Merge two documents

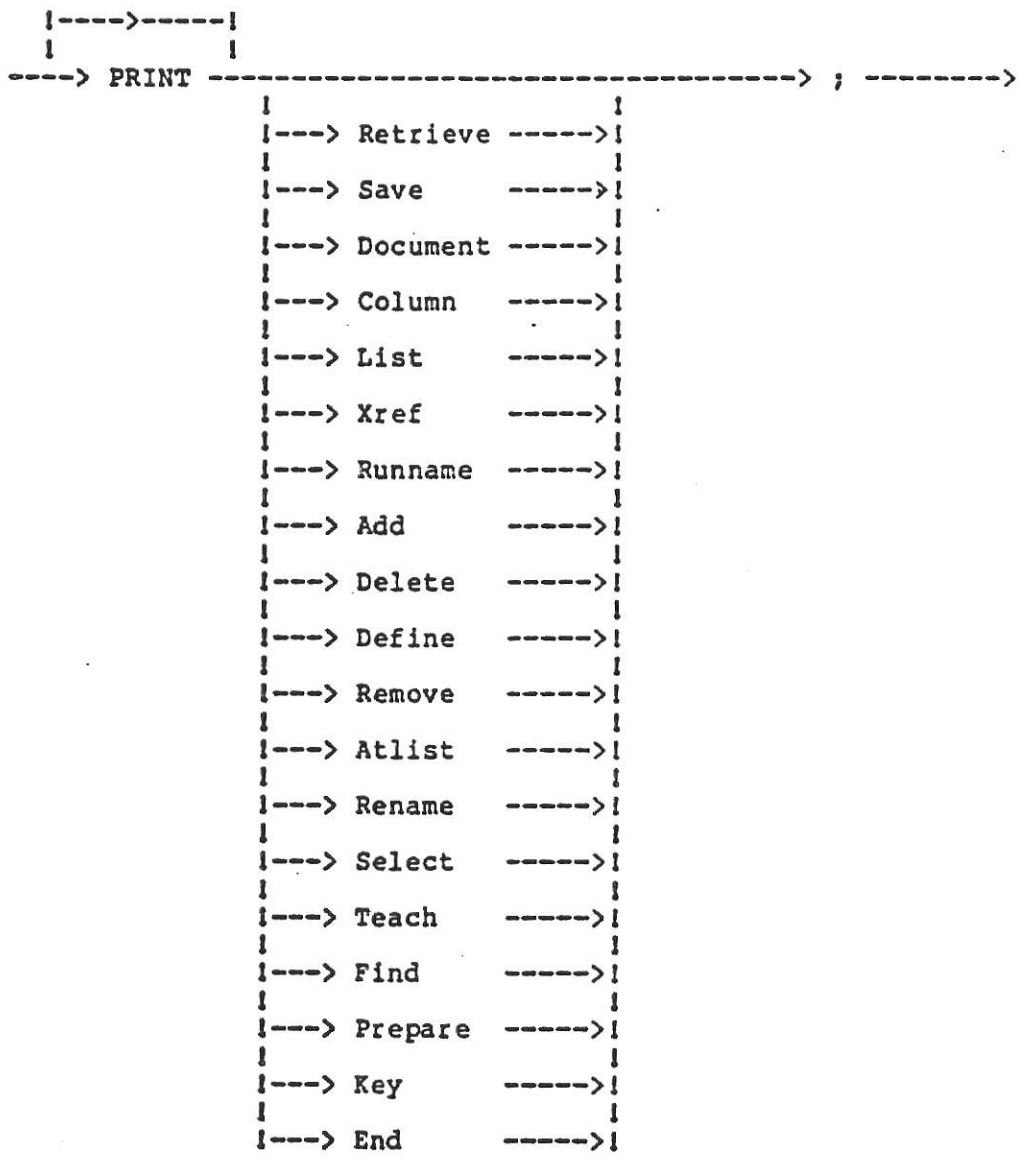
2) Automation of the specification of more than one candidate key
3) define range of attributes
4) Graphical representation of data
5) Specification of relations among the columns
6) Automation of the synonym specification
7) Application to distributed data bases

All these enhancements could be added to the Document Handler but would require additional manpower and storage. However, more experience with the use of the Document Handler is needed. As the users' comments accumulate, overall optimization and enhancements will be planned in detail, thus eliminating excessive modification work.

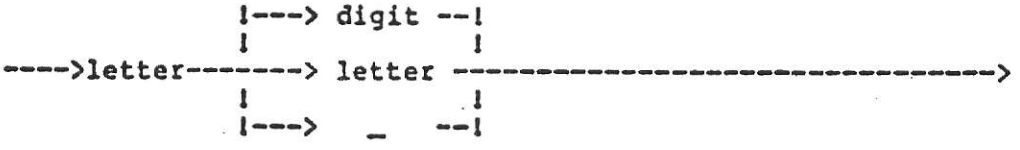
Appendix A

Syntax Graphs

Command :



Name :



Retrieve :

Save :

Document :

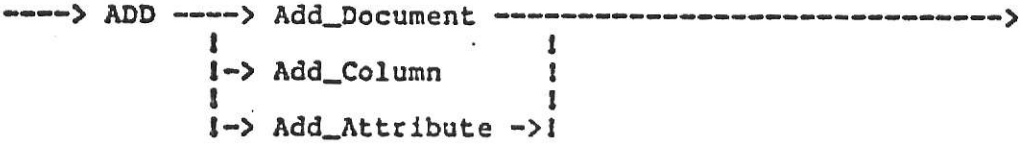
Column :

List :

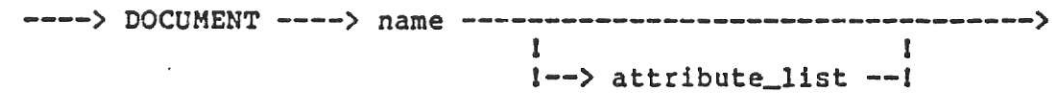
Xref :

Runname :

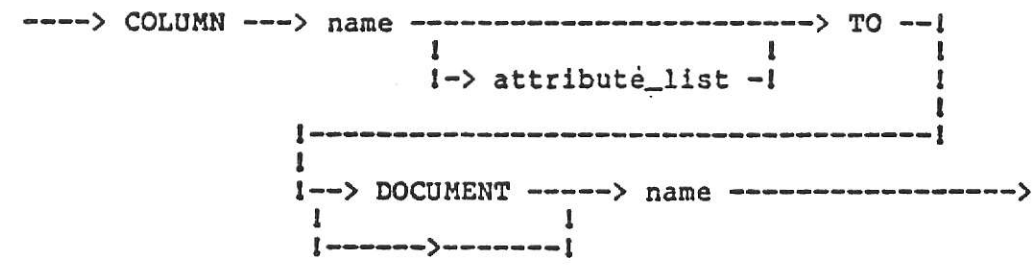
Add :



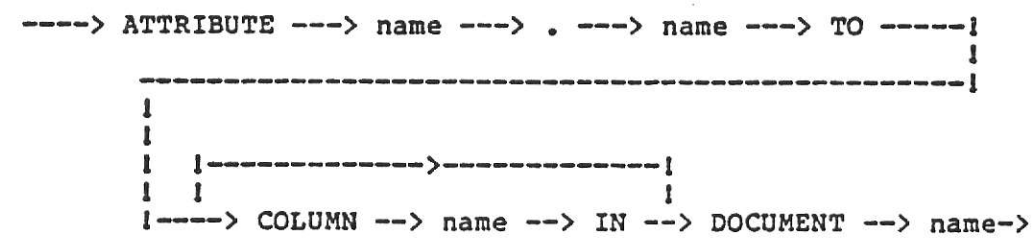
Add_Document :



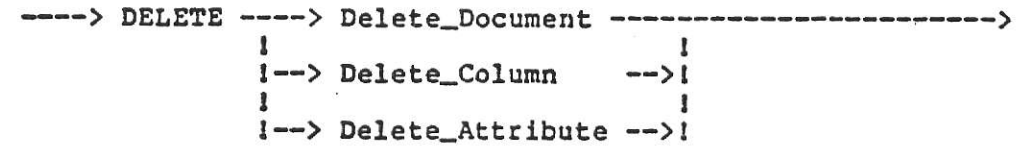
Add_Column :



Add_Attribute :



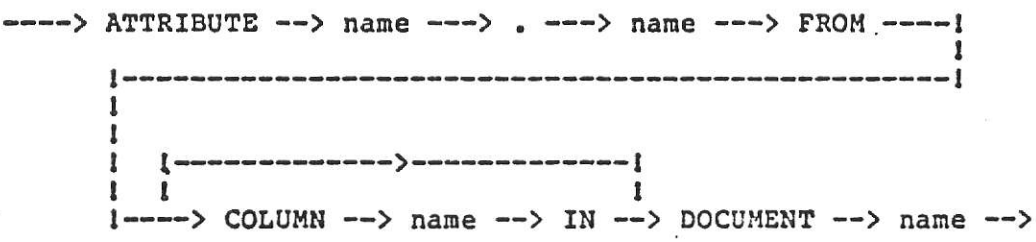
Delete :



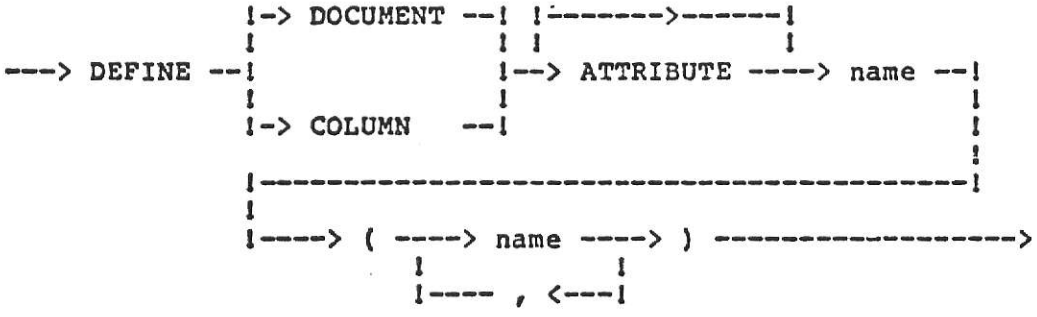
Delete_Document :

Delete_Column :

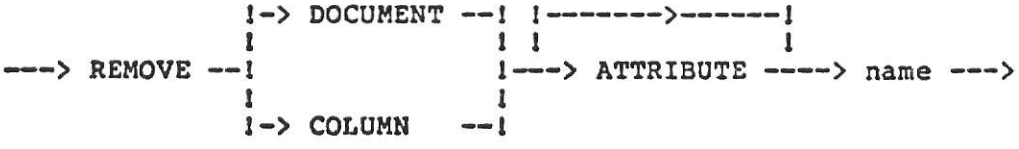
Delete Attribute :



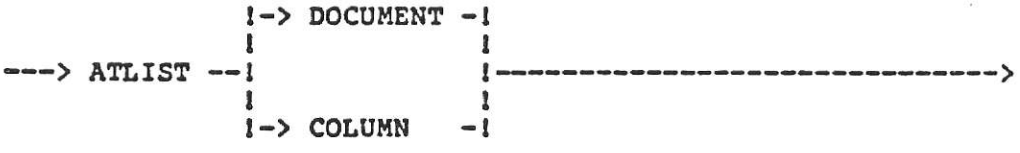
Define :



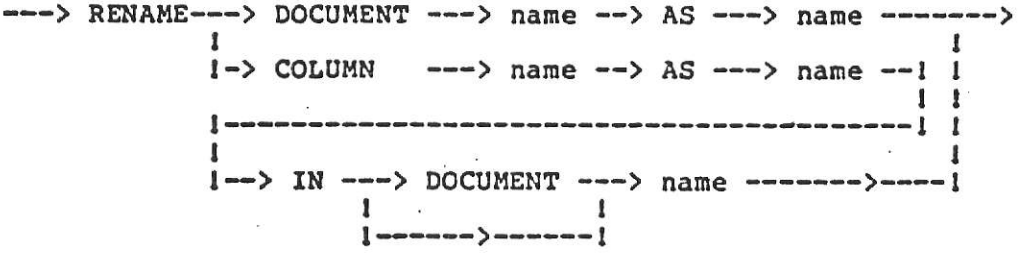
Remove :



Atlist :

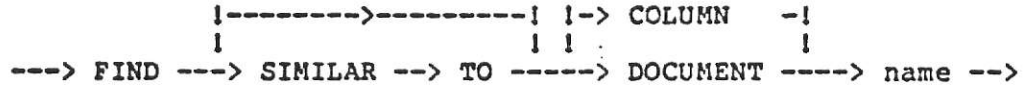


Rename :



Teach :

Find :



Prepare :

Key :

End :

APPENDIX B

DETAILED CODE

```
"MEIR COHEN"
"KANSAS STATE UNIVERSITY"
"DEPARTMENT OF COMPUTER SCIENCE"


"###########
#  PREFIX  #
###########"


CONST NL = '(:10:)';   FF = '(:12:)';   CR = '(:13:)';   EM = '(:25:)';

CONST PAGELENGTH = 512;
TYPE PAGE = ARRAY (.1..PAGELENGTH.) OF CHAR;

CONST LINELENGTH = 132;
TYPE LINE = ARRAY (.1..LINELENGTH.) OF CHAR;

CONST IDLENGTH = 12;
TYPE IDENTIFIER = ARRAY (.1..IDLENGTH.) OF CHAR;

TYPE FILE = 1..2;

TYPE FILEKIND = (EMPTY, SCRATCH, ASCII, SEQCODE, CONCODE);

TYPE FILEATTR = RECORD
                  KIND: FILEKIND;
                  ADDR: INTEGER;
                  PROTECTED: BOOLEAN;
                  NOTUSED: ARRAY (.1..5.) OF INTEGER
                END;

TYPE IODEVICE =
  (TYPEDEVICE, DISKDEVICE, TAPEDEVICE, PRINTDEVICE, CARDDEVICE);

TYPE IOOPERATION = (INPUT, OUTPUT, MOVE, CONTROL);

TYPE IOARG = (WRITEEOF, REWIND, UPSPACE, BACKSPACE);

TYPE IORESULT =
  (COMPLETE, INTERVENTION, TRANSMISSION, FAILURE,
   ENDFILE, ENDMEDIUM, STARTMEDIUM);

TYPE IOPARAM = RECORD
                 OPERATION: IOOPERATION;
                 STATUS: IORESULT;
                 ARG: IOARG
               END;

TYPE TASKKIND = (INPUTTASK, JOBTASK, OUTPUTTASK);

TYPE ARGTAG =
  (NILTYPE, BOOLTYPE, INTTYPE, IDTYPE, PTRTYPE);

TYPE POINTER = @BOOLEAN;
```

```
TYPE PASSPTR = @PASSLINK;

TYPE PASSLINK = RECORD
  OPTIONS: SET OF CHAR;
  FILLER1: ARRAY(.1..7.) OF INTEGER;
  FILLER2: BOOLEAN;
  RESET_POINT: INTEGER;
  FILLER3: ARRAY (.1..11.) OF POINTER
  END;


TYPE ARGTYPE = RECORD
                 CASE TAG: ARGTAG OF
                   NILTYPE, BOOLTYPE: (BOOL: BOOLEAN);
                   INTTYPE: (INT: INTEGER);
                   IDTYPE: (ID: IDENTIFIER);
                   PTRTYPE: (PTR: PASSPTR)
               END;

CONST MAXARG = 10;
TYPE ARGLIST = ARRAY (.1..MAXARG.) OF ARGTYPE;

TYPE ARGSEQ = (INP, OUT);

TYPE PROGRESULT =
  (TERMINATED, OVERFLOW, POINTERERROR, RANGEERROR, VARIANTERROR,
   HEAPLIMIT, STACKLIMIT, CODELIMIT, TIMELIMIT, CALLERROR);


"FILE DESCRIPTOR FOR SVC7 REQUESTS"
TYPE FD_TYPE = PACKED RECORD
       VOLN : PACKED ARRAY [1..4] OF CHAR ;          "VOLUME NAME"
       FN   : PACKED ARRAY [1..8] OF CHAR ;          "FILE NAME"
       EXTN : PACKED ARRAY [1..3] OF CHAR ;          "EXTENSION"
       ACCT : CHAR ;                                 "ACCOUNT NUMBER CODE"
     END;


"SVC 7 PARAMETER BLOCK"

 TYPE SVC7_BLOCK = RECORD
       SVC7_CMD : BYTE ;                    "COMMAND"
       SVC7_MOD : BYTE ;                    "MODIFIER/DEVICE TYPE"
       SVC7_STAT : BYTE ;                   "STATUS"
       SVC7_LU : BYTE ;                     "LOGICAL UNIT NUMBER"
       SVC7_KEYS : SHORTINTEGER ;           "READ/WRITE KEYS"
       SVC7_RECLEN : SHORTINTEGER ;         "LOGICAL RECORD LENGTH"
       SVC7_FD : FD_TYPE ;                  "FILE DESCRIPTOR"
       SVC7_SIZE : INTEGER ;                "FILE(/INDEX) SIZE"
      END;

"SVC 7 COMMAND CODES"

CONST SVC7_ALLOC          = #80 ;
      SVC7_CHAP           = #20 ;
      SVC7_REPORT         = #08 ;
```

```
      SVC7_DELETE         = #02 ;
      SVC7_FETCH_ATTR     = #00 ;
      SVC7_ASSIGN         = #40 ;
      SVC7_RENAME         = #10 ;
      SVC7_CLOSE          = #04 ;
      SVC7_CHECKPT        = #01 ;

"SVC 7 MODIFIER CODES  -  ACCESS PRIVILEGES "

CONST SVC7_AP_SRO     = #00 ;
      SVC7_AP_SWO     = #40 ;
      SVC7_AP_SRW     = #80 ;
      SVC7_AP_ERO     = #20 ;
      SVC7_AP_ERSW    = #C0 ;
      SVC7_AP_EWO     = #60 ;
      SVC7_AP_SREW    = #A0 ;
      SVC7_AP_ERW     = #E0 ;


"SVC 7 MODIFIER CODES  -  BUFFERING/FILE TYPE"

CONST SVC7_BUF_DEFAULT    = #00 ;
      SVC7_BUF_PHYS       = #08 ;
      SVC7_BUF_LOG        = #10 ;
      SVC7_BUF_SVC15      = #18 ;
      SVC7_FTYPE_CONTIG   = #00 ;
      SVC7_FTYPE_CHAIN    = #01 ;
      SVC7_FTYPE_INDEX    = #02 ;
      SVC7_FTYPE_ITAM     = #07 ;


PROCEDURE READ(VAR C: CHAR);
PROCEDURE WRITE(C: CHAR);

PROCEDURE OPEN(F: FILE; ID: IDENTIFIER; VAR FOUND: BOOLEAN);
PROCEDURE CLOSE(F: FILE);
PROCEDURE GET(F: FILE; P: INTEGER; VAR BLOCK: UNIV PAGE);
PROCEDURE PUT(F: FILE; P: INTEGER; VAR BLOCK: UNIV PAGE);
FUNCTION LENGTH(F: FILE): INTEGER;

PROCEDURE MARK(VAR TOP: INTEGER);
PROCEDURE RELEASE(TOP: INTEGER);

PROCEDURE IDENTIFY(HEADER: LINE);
PROCEDURE ACCEPT(VAR C: CHAR);
PROCEDURE DISPLAY(C: CHAR);

PROCEDURE NOTUSED;
PROCEDURE NOTUSED2;
PROCEDURE NOTUSED3;
PROCEDURE NOTUSED4;
PROCEDURE NOTUSED5;
PROCEDURE NOTUSED6;

PROCEDURE NOTUSED7;
```

```
PROCEDURE NOTUSED8;

PROCEDURE NOTUUED9;

PROCEDURE NOTUSED10;

PROCEDURE RUN(ID: IDENTIFIER; VAR PARAM: ARGLIST;
              VAR LINE: INTEGER; VAR RESULT: PROGRESULT);

PROCEDURE RESET ( LU : SHORTINTEGER ) ;


PROGRAM P(PARAM: LINE);
CONST
  NCHAR = 2400 ;
  NAMESIZE = 28 ;
  BLANK8  = '        ' ;
  BLANK28 = '                            ' ;
  LINES_PER_PAGE = 58 ;
  ARRAY_SIZE = 10 ;
  SCREEN_LENGTH = 18 ;
TYPE
  SHORTINT = -32768 .. 32767 ;
  ALFA28 = PACKED ARRAY [ 1 .. NAMESIZE ] OF CHAR ;
  ALFA4 = PACKED ARRAY [ 1 .. 4 ] OF CHAR ;
  ALFA3 = PACKED ARRAY [ 1 .. 3 ] OF CHAR ;
  ALFA60 = PACKED ARRAY [ 1 .. 60 ] OF CHAR ;
  ALFA2 = PACKED ARRAY [ 1 .. 2 ] OF CHAR ;
  ALFA8 = PACKED ARRAY [ 1 .. 8 ] OF CHAR ;
  STATUS = ( WASOK , WASNOTOK ) ;
  IDENT = (BLANK1, PERIOD1, LEFTPAR1, RIGHTPAR1, SEMICOLON1, TEACH1,
           SAVE1, LIST1, XREF1, DOCUMENT1, COLUMN1, FIND1, KEY1,
           SIMILAR1, TO1, PREPARE1, END1, ADD1, RENAME1, ATTRIBUTE1,
           AS1, IN1, DELETE1, DEFINE1, PRINT1, YES1, NO1, NAME1,
           ATLIST1, RETRIEVE1, RUNNAME1, WITH1, WITHOUT1, SELECT1,
           REMOVE1, OTHER1 ) ;
  COMPTR = @COMPONENT ;
  ATTRIBPTR = @ATTRIB ;
  COLPTR = @COLREC ;
  ATTRIB = RECORD
            NAME : ALFA28 ;
            COM : COMPTR ;
            NEXT : ATTRIBPTR
           END;
  COMPONENT = RECORD
               NAME : ALFA28 ;
               NEXT : COMPTR ;
               BACK : ATTRIBPTR
              END ;
  DECLARATION = ( DECLARED , NOTDECLARED ) ;
  ATRPTR = @ATR ;
  ATR    = RECORD
            COMPT : COMPTR ;
            NEXT : ATRPTR ;
          END;
  DOCPTR = @DOCREC ;
```

```
  DOCREC = RECORD
             NAME : ALFA28 ;
             NEXT : DOCPTR ;
             COLU : COLPTR ;
             ATRP : ATRPTR ;
             KEYFD : DECLARATION ;
           END;
  COLOCPTR = @COLOCREC ;
  COLTBPTR = @COLTBREC ;
  COLTBREC = RECORD
               NAME : ALFA28;
               ABV  : ALFA3;
               COLLIST : COLOCPTR;
               NEXT : COLTBPTR;
             END;
  COLOCREC = RECORD
               NEXT : COLOCPTR;
               PDOC : DOCPTR ;
               PCOL : COLPTR ;
             END;
  COLREC = RECORD
             PNAME : COLTBPTR ;
             KEYFD : DECLARATION ;
             ATRP : ATRPTR ;
             NEXT : COLPTR ;
           END;
  ATRARRAY = ARRAY [ 1 .. ARRAY_SIZE ] OF COMPTR ;
VAR
  COMMAND : ARRAY [ 1 .. NCHAR ] OF CHAR;
  LASTCHAR,CURRENT,LINES,PAGES,INDWI,INDWO : SHORTINT ;
  ST : STATUS ;
  LETTERS, DIGITS : SET OF CHAR ;
  WD,NAME_OF_RUN : ALFA28 ;
  COLATFIRST, DOCATFIRST : ATTRIBPTR ;
  DOCTABLE : ARRAY [ 'A' .. 'Z' ] OF DOCPTR ;
  COLTABLE : ARRAY [ 'A' .. 'Z' ] OF COLTBPTR ;
  CH : CHAR ;
  PRINTSW,RETRIEVESW,WAS_A_RT,WAS_A_SV,DELTSW : BOOLEAN ;
  DATE : ALFA8 ;
  WITHAR, WITHOUTAR : ATRARRAY ;


PROCEDURE SVC7(VAR PARM:SVC7_BLOCK); EXTERN ;
PROCEDURE SVC2FDAT(VAR MMDDYY:ALFA8); EXTERN ;
PROCEDURE SVC2PAUS ; EXTERN ;

PROCEDURE NEXTSYM ( VAR WD : ALFA28 ; VAR CODE : IDENT ) ; FORWARD ;
PROCEDURE DISP28NL ( W : ALFA28 ) ; FORWARD ;
PROCEDURE GETCOMD ( VAR S : STATUS ) ; FORWARD ;
PROCEDURE SEARCHDOC(W:ALFA28;VAR FOUND:BOOLEAN;VAR P1:DOCPTR);FORWARD;
PROCEDURE ADD ; FORWARD ;
PROCEDURE DEFINE ; FORWARD ;
PROCEDURE KEY ; FORWARD ;
PROCEDURE DISP8(W:ALFA8); FORWARD;
PROCEDURE DISP28(W:ALFA28) ; FORWARD ;
PROCEDURE DISP28BL ( W : ALFA28 ) ; FORWARD ;
```

```
PROCEDURE WRITE28 ( W : ALFA28 ) ; FORWARD ;
PROCEDURE WRITE8 ( W : ALFA8 ) ; FORWARD ;
PROCEDURE WRITE3 ( W : ALFA3 ) ; FORWARD ;
PROCEDURE WRITE2 ( W : ALFA2 ) ; FORWARD ;
PROCEDURE BLANKS ( N : SHORTINT ) ; FORWARD ;
PROCEDURE NEXT_SCREEN ; FORWARD ;


(*****************************************************************************)
(*                                                                           *)
(*                      O U T P U T    R O U T I N E S                       *)
(*                                                                           *)
(*****************************************************************************)


PROCEDURE RUNNAME ;
CONST
  ERRORMES = '**** RUNNAME - SYNTAX ERROR ' ;
VAR
  CODE : IDENT ;
  W    : ALFA28 ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF CODE = SEMICOLON1 THEN
   BEGIN
    DISP8('RUN NAME') ;
    DISPLAY(' '); DISP28NL(NAME_OF_RUN);
   END
  ELSE
  IF CODE <> BLANK1
   THEN
    DISP28NL(ERRORMES)
   ELSE
    BEGIN
     NEXTSYM ( W , CODE ) ;
     IF CODE <> NAME1
      THEN
       DISP28NL(ERRORMES)
      ELSE
       NAME_OF_RUN := W ;
    END;
END;


PROCEDURE CHARINT ( N : SHORTINT ; VAR W : ALFA4 ) ;
VAR
  N1, N2, I  : SHORTINT ;
  DIG : ARRAY [ 0 .. 9 ] OF CHAR ;
BEGIN
  IF ( N > 9999 ) OR ( N < 0 )
   THEN W := '****'
   ELSE
    BEGIN
     N1 := 10000 ;
     N2 := N ;
     DIG[0] := '0'; DIG[1] := '1'; DIG[2] := '2'; DIG[3] := '3'; DIG[4] :=
```

```
'4';
     DIG[5] := '5'; DIG[6] := '6'; DIG[7] := '7'; DIG[8] := '8'; DIG[9] :=
'9';
     FOR I := 1 TO 4 DO
      BEGIN
       N1 := N1 DIV 10 ;
       W [ I ] := DIG [ N2 DIV N1 ] ;
       N2 := N2 MOD N1 ;
      END ;
     I := 1 ;
     WHILE ( W[I] = '0' ) AND ( I<4 ) DO
      BEGIN
       W [ I ] := ' ' ;
       I := I + 1 ;
      END ;
    END ;
END;


PROCEDURE WRITEINT ( N : SHORTINT ) ;
VAR
  W : ALFA4 ;
  I : SHORTINT ;
BEGIN
  CHARINT ( N , W ) ;
  FOR I := 1 TO 4 DO
   WRITE ( W [ I ] ) ;
END;


PROCEDURE DISPINT ( N : SHORTINT ) ;
VAR
  W : ALFA4 ;
  I : SHORTINT ;
BEGIN
  CHARINT ( N , W ) ;
  FOR I := 1 TO 4 DO
   DISPLAY ( W [ I ] ) ;
END;


PROCEDURE HEADER ;
VAR
  I : SHORTINT ;
BEGIN
  WRITE ( FF ) ;
  PAGES := PAGES + 1 ;
  WRITE28(' DOCUMENT   HANDLER    VER ');
  WRITE3('0.0');BLANKS(8);WRITE3('RUN');BLANKS(2);
  WRITE28(NAME_OF_RUN);BLANKS(8);
  WRITE8('  DATE  '); WRITE8(DATE);
  BLANKS(19); WRITE8(' PAGE   '); WRITEINT(PAGES);
  WRITE(NL);
  FOR I := 1 TO 65 DO
   BEGIN
    WRITE('-'); WRITE(' ');
```

```
   END;
  WRITE(NL); WRITE(NL);
  LINES := 4 ;
END;


PROCEDURE NEW_LINE ;
BEGIN
  LINES := LINES + 1 ;
  WRITE ( NL ) ;
  IF LINES > LINES_PER_PAGE THEN
    HEADER ;
END;


PROCEDURE DISP8 ;
VAR I : 1 .. 8 ;
BEGIN
  FOR I := 1 TO 8 DO
            DISPLAY ( W [ I ] )
END;


PROCEDURE DISP60NL ( W : ALFA60 ) ;
VAR
  I : SHORTINT ;
BEGIN
  FOR I := 1 TO 60 DO
   DISPLAY ( W [ I ] ) ;
  DISPLAY ( NL ) ;
END;


PROCEDURE DISP28 ;
VAR I : 1..NAMESIZE ;
BEGIN
  FOR I := 1 TO NAMESIZE DO
            DISPLAY( W [ I ] )
END;


PROCEDURE DISP28NL ;
BEGIN
  DISP28 ( W ) ;
  DISPLAY ( NL )
END;


PROCEDURE DISP28BL ;
VAR I : 0 .. NAMESIZE ;
BEGIN
  I := 0 ;
  REPEAT
    I := I + 1 ;
    DISPLAY ( W [ I ] ) ;
  UNTIL ( W[I]=' ')OR(I=NAMESIZE )
```

```
END;


PROCEDURE WRITE28 ;
VAR I : 1 .. NAMESIZE ;
BEGIN
  FOR I := 1 TO NAMESIZE DO
    WRITE ( W[I] ) ;
END ;


PROCEDURE WRITE28BL ( W : ALFA28 ) ;
VAR
  I : 0 .. NAMESIZE ;
BEGIN
  I := 0 ;
  REPEAT
   I := I + 1 ;
   WRITE ( W [ I ] ) ;
  UNTIL ( W [ I ] = ' ' ) OR ( I = NAMESIZE ) ;
END ;


PROCEDURE BLANKS ;
VAR
  I : SHORTINT ;
BEGIN
  IF N > 0 THEN
   FOR I := 1 TO N DO
    WRITE (' ') ;
END;


PROCEDURE WRITE3 ;
BEGIN
  WRITE ( W [ 1 ] ) ;
  WRITE ( W [ 2 ] ) ;
  WRITE ( W [ 3 ] ) ;
END;


PROCEDURE WRITE2 ;
BEGIN
  WRITE ( W [ 1 ] ) ;
  WRITE ( W [ 2 ] ) ;
END;


PROCEDURE WRITE8 ;
VAR I : 1 .. 8 ;
BEGIN
  FOR I := 1 TO 8 DO
    WRITE ( W [I] ) ;
END ;
```

```
PROCEDURE DOCNOTFOUND ( W : ALFA28 ) ;
BEGIN
  DISP8('DOCUMENT'); DISPLAY(' '); DISP28BL(W) ;
  DISP8(' NOT FOU'); DISPLAY('N'); DISPLAY('D');
  DISPLAY(NL);
END;


PROCEDURE COLNOTFOUND(W1,W2:ALFA28);
BEGIN
  DISP8('COLUMN  '); DISP28BL(W1);
  DISP28(' DOES NOT EXIST IN DOCUMENT ');DISP28BL(W2);
  DISPLAY(NL);
END;


(*****************************************************************************)
(*                                                                           *)
(*                    T E A C H          R O U T I N E S                     *)
(*                                                                           *)
(*****************************************************************************)


PROCEDURE TEACHPE ;
BEGIN
  DISP60NL('*PREPARE* OR *PE* PREPARES A FILE WITH FUNCTIONAL-DEPEND-');
  DISP60NL('ENCIES FOR BERNSTEIN"S ALGORYTHM USING ABBREVIATED NAMES.');
  DISP60NL('A DICTIONARY OF THE COLUMN NAMES VS. THE ABBREVIATED ONES');
  DISP60NL('IS WRITTEN ON LOGICAL UNIT 2.                             ');
  DISP60NL('ABBREVIATED : PE                                          ');
  DISP60NL('CALLING FORM : PREPARE ;                                  ');
END;


PROCEDURE TEACHCL ;
BEGIN
  DISP60NL('*COLUMN* OR *CL* WILL LIST ALL THE COLUMNS IN THE SYSTEM. ');
  DISP60NL('IF PRECEDED BY PRINT, THE LISTING WILL BE DONE ON THE     ');
  DISP60NL('DEVICE OR FILE ASSOCIATED WITH LOGICAL UNIT 2.            ');
  DISP60NL('ABBREVIATION : CL                                         ');
  DISP60NL('CALLING FORM : COLUMN ;                                   ');
END;


PROCEDURE TEACHPR ;
BEGIN
  DISP60NL('*PRINT* OR *PR* WHEN PRECEDING A COMMAND WILL CAUSE THE   ');
  DISP60NL('OUTPUT TO BE DIVERTED TO THE PRINTER (OR ANY OTHER DEVICE ');
  DISP60NL('OR FILE ASSOCIATED WITH UNIT 2) .                         ');
  DISP60NL('ABBREVIATION : PR                                         ');
END ;


PROCEDURE TEACHRN ;
BEGIN
  DISP60NL('*RENAME* OR *RN* WILL RENAME A DOCUMENT OR A COLUMN WITHIN');
```

```
  DISP60NL('A DOCUMENT.                                                ');
  DISP60NL('ABBREVIATION : RN                                          ');
  DISP60NL('CALLING FORM :                                             ');
  DISP60NL(' RENAME DOCUMENT <NAME> AS <NAME> ;                        ');
  DISP60NL(' RENAME COLUMN <NAME> AS <NAME> IN DOCUMENT <NAME> ;       ');
END ;


PROCEDURE TEACHAL ;
BEGIN
  DISP60NL('*ATLIST* OR *AL* WILL LIST ALL THE DOCUMENT OR COLUMN      ');
  DISP60NL('ATTRIBUTES THAT ARE DEFINED IN THE SYSTEM.                 ');
  DISP60NL('ABBREVIATION : AL                                          ');
  DISP60NL('CALLING FORM : ATLIST DOCUMENT ;                           ');
  DISP60NL('               ATLIST COLUMN ;                             ');
  DISP60NL('               ATLIST DOCUMENT <NAME>,<NAME>,...           ');
  DISP60NL('               ATLIST COLUMN <NAME>,<NAME>,...             ');
END;


PROCEDURE TEACHRM ;
BEGIN
  DISP60NL('*REMOVE* OR *RM* WILL DELETE A DOCUMENT OR A COLUMN        ');
  DISP60NL('ATTRIBUTE WITH ALL ITS OCCURANCES IN THE SYSTEM.           ');
  DISP60NL('ABBREVIATION : RM                                          ');
  DISP60NL('CALLING FORM : REMOVE DOCUMENT ATTRIBUTE <NAME> ;          ');
  DISP60NL('               REMOVE COLUMN ATTRIBUTE <NAME> ;            ');
END;


PROCEDURE TEACHRT ;
BEGIN
 DISP60NL('*RETRIEVE* OR *RT* WILL LOAD THE ENTIRE STRUCTURE FROM UNIT');
 DISP60NL('2. RETRIEVE IS PERMITTED ONLY ONCE IN A SESSION, BUT IT    ');
 DISP60NL('DOES NOT HAVE TO BE THE FIRST COMMAND.                     ');
 DISP60NL('ABBREVIATION : RT                                          ');
 DISP60NL('CALLING FORM : RETRIEVE                                    ');
END;


PROCEDURE TEACHFI ;
BEGIN
  DISP60NL('*FIND* OR *FI* LISTS ALL NAMES THAT HAVE UPTO TWO SPELLING');
  DISP60NL('DIFFERENCES FROM THE PARAMETER IN THE CALL.                ');
  DISP60NL('ABBREVIATION : FI                                          ');
  DISP60NL('CALLING FORM : FIND SIMILAR TO DOCUMENT <NAME> ;           ');
  DISP60NL('               FIND DOCUMENT <NAME> ;                      ');
  DISP60NL('               FIND SIMILAR TO COLUMN <NAME> ;             ');
  DISP60NL('               FIND COLUMN <NAME> ;                        ');
END;


PROCEDURE TEACHLS ;
BEGIN
  DISP60NL('*LIST* OR *LS* GIVES A LIST OF ALL THE DOCUMENTS IN THE    ');
  DISP60NL('SYSTEM WITH THEIR COLUMNS AND ATTRIBUTES. IF PRECEDED BY A ');
```

```
  DISP60NL('PRINT COMMAND, THE LISTING WILL BE DIVERTED TO THE PRINTER.');
  DISP60NL('ABBREVIATION : LS                                          ');
  DISP60NL('CALLING FORM : LIST ;                                      ');
  DISP60NL('               LIST <NAME>                                 ');
END;


PROCEDURE TEACHXR;
BEGIN
  DISP60NL(' *XREF* OR *XR* GIVES A CROSS-REFERENCE OF THE COLUMNS IN ');
  DISP60NL(' THE SYSTEM, THAT IS FOR EACH COLUMN NAME, A LIST OF ALL  ');
  DISP60NL(' THE DOCUMENTS WHERE THIS COLUMN NAME OCCURS IS GIVEN.    ');
  DISP60NL(' ABBREVIATION : XR                                        ');
  DISP60NL(' CALLING FORM : XREF ;                                    ');
  DISP60NL('                XRREF <NAME>                              ');
END;


PROCEDURE TEACHSV ;
BEGIN
  DISP60NL(' *SAVE* OR *SV* SAVES THE ENTIRE STRUCTURE FOR FURTHER    ');
  DISP60NL('USE.                                                      ');
  DISP60NL('SAVED DATA IS WRITTEN ON UNIT 2 AND IT IS THE RESPONSIBI- ');
  DISP60NL('LITY OF THE USER TO ASSIGN A FILE TO UNIT 2 .              ');
  DISP60NL('ABBREVIATION : SV                                         ');
  DISP60NL('CALLING FORM : SAVE ;                                     ');
END;


PROCEDURE TEACHDC ;
BEGIN
  DISP60NL(' *DOCUMENT* OR *DC* WILL LIST ALL DOCUMENTS IN THE SYSTEM.');
  DISP60NL(' IF PRECEDED BY PRINT, THE LISTING WILL BE DONE ON THE    ');
  DISP60NL(' DEVICE OR FILE ASSOCIATED WITH LOGICAL UNIT 2.           ');
  DISP60NL(' ABBREVIATION : DC                                        ');
  DISP60NL(' CALLING FORM : DOCUMENT ;                                ');
END;


PROCEDURE TEACHAD ;
BEGIN
  DISP60NL('*ADD* OR *AD* ADDS A DOCUMENT TO THE STRUCTURE AND/OR     ');
  DISP60NL('A COLUMN TO A DOCUMENT AND/OR AN ATTRIBUTE TO A COLUMN OR ');
  DISP60NL('A DOCUMENT.                                               ');
  DISP60NL('ABBREVIATION : AD                                         ');
  DISP60NL('CALLING FORM :                                            ');
  DISP60NL(' ADD DOCUMENT <NAME>;                                     ');
  DISP60NL(' ADD COLUMN <NAME> TO DOCUMENT <NAME>;                    ');
  DISP60NL(' ADD ATTRIBUTE <NAME>.<NAME> TO DOCUMENT <NAME>;          ');
  DISP60NL(' ADD ATTRIBUTE <NAME>.<NAME> TO COLUMN <NAME> IN DOCUMENT ');
  DISP60NL('                                          <NAME>;  ');
  DISP60NL(' IN THE FIRST TWO FORMS ANY COLUMN OR DOCUMENT NAME MAY BE');
  DISP60NL('FOLLOWED BY A LIST OF ATTRIBUTES IN PARENTHESES.          ');
  DISP60NL(' EXAMPLE:  AD CL DATE(LOC.INP,FREQUENCY.DAILY)TO DC RECEPT');
END;
```

```
PROCEDURE TEACHDL ;
BEGIN
  DISP60NL('*DELETE* OR *DL* DELETES A DOCUMENT FROM THE STRUCTURE AND');
  DISP60NL('/OR A COLUMN FROM A DOCUMENT AND/OR AN ATTRIBUTE FROM A   ');
  DISP60NL('DOCUMENT OR A COLUMN.                                     ');
  DISP60NL('ABBREVIATION : DL                                         ');
  DISP60NL('CALLING FORM :                                            ');
  DISP60NL(' DELETE DOCUMENT <NAMME> ;                                ');
  DISP60NL(' DELETE COLUMN <NAME> FROM DOCUMENT <NAME> ;              ');
  DISP60NL(' DELETE ATTRIBUTE <NAME>.<NAME> FROM DOCUMENT <NAME> ;    ');
  DISP60NL(' DELETE ATTRIBUTE <NAME>.<NAME> FROM COLUMN <NAME> IN     ');
  DISP60NL('                                        DOCUMENT <NAME> ; ');
END;


PROCEDURE TEACHRU ;
BEGIN
  DISP60NL('*RUNNAME* OR *RU* IS USED TO ESTABLISH A NAME FOR EACH RUN');
  DISP60NL('OR EACH MODEL. THIS NAME, ONCE IT WAS ESTABLISHED IS SAVED');
  DISP60NL('ON THE SAVED DATA FILE.                                   ');
  DISP60NL('ABBREVIATION : RU                                         ');
  DISP60NL('CALLING FORM: RUNNAME ;                                   ');
  DISP60NL('              RUNNAME <NAME> ;                            ');
  DISP60NL('THE FIRST FORM WILL DISPLAY THE CURRENT RUN NAME, THE     ');
  DISP60NL('SECOND FORM ESTABLISHES A NEW NAME .                      ');
END;


PROCEDURE TEACHTE ;
BEGIN
  DISP60NL('YOU ARE IN TEACH AT THIS VERY MOMENT !                    ');
END;


PROCEDURE TEACHLP ( VAR CODE : IDENT ) ;
VAR S : STATUS ;
    W : ALFA28 ;
BEGIN
  DISP60NL(' THE FOLLOWING COMMANDS ARE SUPPORTED BY THIS SYSTEM      ');
  DISP60NL('                                                          ');
  DISP60NL(' ADD    (AD)       ATLIST   (AL)       KEY     (KY)     ');
  DISP60NL(' DELETE (DL)       LIST     (LS)       PREPARE (PE)     ');
  DISP60NL(' RENAME (RN)       DOCUMENT (DC)       XREF    (XR)     ');
  DISP60NL(' DEFINE (DF)       COLUMN   (CL)       FIND    (FI)     ');
  DISP60NL(' SAVE   (SV)       PRINT    (PR)       END     (EN)     ');
  DISP60NL(' TEACH  (TE)       RETRIEVE (RT)       RUNNAME (RU)     ');
  DISP60NL(' REMOVE (RM)       SELECT   (SL)                        ');
  DISP60NL('                                                        ');
  DISP60NL(' TO GET AN EXPLANATION AND CALLING FORM, TYPE THE COMMAND ');
  DISP60NL(' NAME, TO EXIT TEACH MODE, TYPE END. TO RETURN TO MTM TYPE');
  DISP60NL(' ANOTHER END. FURTHER INFORMATION CAN BE FOUND IN THE USER');
  DISP60NL(' MANUAL                                                 ');
  GETCOMD ( S ) ;
  NEXTSYM ( W , CODE ) ;
  IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
```

```
END;


PROCEDURE TEACH ;
VAR
  CODE : IDENT ;
  S : STATUS ;
BEGIN
  REPEAT
  TEACHLP ( CODE ) ;
  CASE CODE OF
   DOCUMENT1 : TEACHDC ;
   COLUMN1   : TEACHCL ;
   ADD1      : TEACHAD ;
   DELETE1   : TEACHDL ;
   RENAME1   : TEACHRN ;
   DEFINE1   : ;
   SAVE1     : TEACHSV ;
   ATLIST1   : TEACHAL ;
   LIST1     : TEACHLS ;
   PRINT1    : TEACHPR ;
   KEY1      : ;
   PREPARE1  : TEACHPE ;
   XREF1     : TEACHXR ;
   FIND1     : TEACHFI ;
   TEACH1    : TEACHTE ;
   END1      : ;
   RETRIEVE1 : TEACHRT ;
   REMOVE1   : TEACHRM ;
   RUNNAME1  : TEACHRU ;
   SELECT1   : ;
   ELSE      : DISP28NL('WRONG COMMAND NAME          ');
  END;
  DISPLAY(NL); IF CODE<>END1 THEN GETCOMD(S);
  UNTIL CODE = END1 ;
END;


(*****************************************************************************)
(*                                                                           *)
(*             S A V E                    R O U T I N E S                *)
(*                                                                           *)
(*****************************************************************************)


PROCEDURE SVATR ( FIRST : ATTRIBPTR ; AL : ALFA2 ) ;
VAR
  T1 : ATTRIBPTR ;
  R1 : COMPTR ;
BEGIN
  T1 := FIRST ;
  WHILE T1 <> NIL DO
   BEGIN
   WRITE2('DF'); WRITE(' ');
   WRITE2( AL ); WRITE(' ');
   WRITE2('AT'); WRITE(' ');
```

```
    WRITE28(T1@.NAME);
    R1 := T1@.COM ;
    WRITE('(');
    WHILE R1<>NIL DO
     BEGIN
      WRITE(NL); WRITE28(R1@.NAME);
      R1 := R1@.NEXT ;
     END;
    WRITE(')'); WRITE(';'); WRITE(NL);
    T1 := T1@.NEXT ;
    END
END;


PROCEDURE SVKEY ( P : DOCPTR ) ;
VAR
  R : COLPTR ;
BEGIN
  WRITE3('KEY'); WRITE(' ');
  WRITE28(P@.NAME); WRITE(';');
  WRITE(NL);
  R := P@.COLU ;
  WHILE R <> NIL DO
   BEGIN
    IF R@.KEYFD = DECLARED
     THEN
      BEGIN
       WRITE28(R@.PNAME@.NAME) ;
       WRITE ( NL ) ;
      END;
    R := R@.NEXT ;
   END;
  WRITE(';'); WRITE(NL) ;
END;


PROCEDURE SVATRP ( S1 : ATRPTR ) ;
VAR
  S : ATRPTR ;
BEGIN
  IF S1 <> NIL THEN
   BEGIN
    S := S1 ;
    WRITE ('(') ;
     WHILE S<>NIL DO
     BEGIN
      WRITE28(S@.COMPT@.BACK@.NAME);
       WRITE('.') ;
      WRITE28(S@.COMPT@.NAME);
      WRITE ( NL ) ;
      S := S@.NEXT ;
     END ;
    WRITE ( ')' ) ;
   END
END ;
```

```
PROCEDURE SVDOC ( P : DOCPTR ) ;
VAR
  Q : COLPTR ;
BEGIN
  WRITE2('AD'); WRITE(' ');
  WRITE2('DC'); WRITE(' ');
  WRITE28 ( P@.NAME) ;
  SVATRP ( P@.ATRP ) ;
  WRITE(';'); WRITE(NL);
  Q := P@.COLU ;
  IF Q <> NIL THEN
   BEGIN
    WHILE Q <> NIL DO
     BEGIN
      WRITE28(Q@.PNAME@.NAME); WRITE(NL);
      SVATRP ( Q@.ATRP ) ;
      Q := Q@.NEXT ;
     END;
    WRITE(';'); WRITE(NL);
   END;
END;


PROCEDURE SAVE ;
VAR
  CH : CHAR ;
  P  : DOCPTR ;
  BLK0, BLK1, BLK2, BLK3 : SVC7_BLOCK ;
BEGIN
  WITH BLK0 DO
   BEGIN
    SVC7_CMD := SVC7_CLOSE ;
    SVC7_MOD := SVC7_BUF_DEFAULT ;
    SVC7_LU  := 1 ;
   END ;
  WITH BLK1 DO
   BEGIN
    SVC7_CMD := SVC7_CLOSE ;
    SVC7_MOD := SVC7_BUF_DEFAULT ;
    SVC7_LU  := 2 ;
   END ;
  WITH BLK2 DO
   BEGIN
    SVC7_CMD := SVC7_ASSIGN ;
    SVC7_MOD := SVC7_AP_SRW ;
    SVC7_LU  := 2 ;
    SVC7_KEYS := 0 ;
    SVC7_RECLEN := 126 ;
    SVC7_FD.VOLN := 'USR6' ;
    SVC7_FD.FN   := 'DUMMY   ' ;
    SVC7_FD.EXTN := 'LIS' ;
    SVC7_FD.ACCT := 'P' ;
   END;
  WITH BLK3 DO
   BEGIN
```

```
    SVC7_CMD := SVC7_ASSIGN ;
    SVC7_MOD := SVC7_BUF_DEFAULT ;
    SVC7_LU  := 2 ;
    SVC7_FD.VOLN := 'PR  ' ;
    SVC7_FD.FN   := '        ' ;
    SVC7_FD.EXTN := '    ' ;
    SVC7_FD.ACCT := ' ' ;
   END;
  SVC7 ( BLK0 ) ;
  SVC7 ( BLK1 ) ;
  SVC7 ( BLK2 ) ;
  RESET ( 2 ) ;
  IF NAME_OF_RUN <> BLANK28
   THEN
    BEGIN
     WRITE8('RUNNAME '); WRITE28(NAME_OF_RUN);
     WRITE(';'); WRITE(NL);
    END;
  SVATR ( DOCATFIRST , 'DC' ) ;
  SVATR ( COLATFIRST , 'CL' ) ;
  FOR CH := 'A' TO 'Z' DO
   BEGIN
    P := DOCTABLE [ CH ] ;
    WHILE P <> NIL DO
     BEGIN
      SVDOC ( P ) ;
      IF P@.KEYFD = DECLARED THEN SVKEY ( P ) ;
      P := P@.NEXT ;
     END
   END ;
  WRITE3('END'); WRITE(';'); WRITE(NL);
  SVC7 ( BLK1 ) ;
  SVC7 ( BLK3 ) ;
  RESET ( 2 ) ;
  WAS_A_SV := TRUE ;
END;


(*****************************************************************************)
(*                                                                           *)
(*            R E T R I E V E          R O U T I N E S                       *)
(*                                                                           *)
(*****************************************************************************)


PROCEDURE GETCOMD1;
VAR
  CH : CHAR ;
  I  : SHORTINT ;
BEGIN
  CURRENT := 1 ;
  LASTCHAR := 0 ;
  REPEAT
    READ ( CH ) ;
    CASE CH OF
  NL,',' : BEGIN
```

```
            LASTCHAR := LASTCHAR + 1 ;
            COMMAND [ LASTCHAR ] := ' ' ;
           END;
     ELSE : BEGIN
            LASTCHAR := LASTCHAR + 1 ;
            COMMAND [ LASTCHAR ] := CH ;
           END;
    END;
  UNTIL(CH=';') OR (LASTCHAR >= NCHAR);
  IF LASTCHAR < NCHAR
   THEN
     FOR I := LASTCHAR + 1 TO NCHAR DO
      COMMAND [ I ] := ';'
   ELSE
     DISP28NL('ERROR - COMMAND IS TOO LONG ');
  READ ( CH ) ;
END;


PROCEDURE RETRIEVE ;
VAR
  CH : CHAR ;
  W : ALFA28 ;
  CODE : IDENT ;
BEGIN
  IF WAS_A_RT THEN
    DISP28NL('RETRIEVE NOT PERMITTED      ')
  ELSE
   BEGIN
  RETRIEVESW := TRUE ;
  GETCOMD1 ;
  NEXTSYM ( W , CODE ) ;
  WHILE CODE <> END1 DO
   BEGIN
    CASE CODE OF
     BLANK1  : ;
     RUNNAME1 : RUNNAME ;
     ADD1    : ADD ;
     DEFINE1 : DEFINE ;
     KEY1    : KEY ;
     SEMICOLON1,END1    : ;
    END;
    GETCOMD1 ;
    NEXTSYM ( W , CODE ) ;
   END;
  RETRIEVESW := FALSE ;
  WAS_A_RT := TRUE;
 END;
END;

(*************************************************************************)
(*                                                                       *)
(*           L I S T                  R O U T I N E S                    *)
(*                                                                       *)
(*************************************************************************)
```

```
PROCEDURE LISTDOC ( P : DOCPTR ) ;
  (* LIST DOCUMENT POINTED BY P *)
VAR
  Q : COLPTR ;
  R : ATRPTR ;
BEGIN
  WITH P@ DO
   IF PRINTSW
    THEN
     BEGIN
      NEW_LINE  ;
      BLANKS ( 8 ) ;
      IF KEYFD = DECLARED THEN WRITE('*') ELSE WRITE(' ');
      WRITE (' ');
      WRITE28 (NAME) ; NEW_LINE ;
       IF ATRP <> NIL THEN
        BEGIN
         BLANKS ( 38 ) ;
         WRITE28('DOCUMENT  ATTRIBUTES  :     ');
         NEW_LINE ;
         R := ATRP ;
         WHILE R <> NIL DO
          BEGIN
           BLANKS ( 46 ) ;
           WRITE28BL ( R@.COMPT@.BACK@.NAME ) ;
           WRITE2 ( '. ' ) ;
           WRITE28BL ( R@.COMPT@.NAME ) ;
           NEW_LINE ; R := R @.NEXT ;
          END ;
      END
    END
    ELSE
     BEGIN
      IF KEYFD=DECLARED THEN DISPLAY('*') ELSE DISPLAY(' ');
      DISP28NL ( NAME );
     END;
  Q := P@.COLU ;
  WHILE Q <> NIL DO
   WITH Q@ DO
    BEGIN
     IF PRINTSW
      THEN
       BEGIN
        BLANKS ( 24 );
        IF KEYFD=DECLARED THEN WRITE('*') ELSE WRITE(' ');
        WRITE(' ');
        WRITE28(PNAME@.NAME); NEW_LINE ;
        IF ATRP <> NIL THEN
         BEGIN
          R := ATRP ;
          WHILE R <> NIL DO
           BEGIN
            BLANKS ( 30 ) ;
            WRITE28BL(R@.COMPT@.BACK@.NAME);
```

```
            WRITE2('. ');
            WRITE28BL(R@.COMPT@.NAME);
            NEW_LINE ;
            R := R@.NEXT
           END
         END
       END
      ELSE
       BEGIN
        DISP8(BLANK8);
        IF KEYFD=DECLARED THEN DISPLAY('*')ELSE DISPLAY(' ');
        DISP28NL(PNAME@.NAME)
       END;
     Q := NEXT ;
    END;
END;


PROCEDURE LIST ;
CONST ERRORMES='**** LIST - SYNTAX ERROR    ' ;
VAR
  W : ALFA28 ;
  CODE : IDENT ;
  CH : CHAR ;
  P : DOCPTR ;
  FOUND : BOOLEAN ;
  S : STATUS ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF (CODE<>BLANK1)AND(CODE<>SEMICOLON1) THEN DISP28NL(ERRORMES)
 ELSE BEGIN
  IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
  IF PRINTSW THEN
    BEGIN
     HEADER ;  BLANKS ( 44 ) ;
     WRITE28 ('DOCUMENT / COLUMN  LISTING  '); NEW_LINE ;
    END;
  IF CODE = SEMICOLON1
   THEN
    FOR CH := 'A' TO 'Z' DO
     BEGIN
      P := DOCTABLE [ CH ] ;
      WHILE P <> NIL DO
       BEGIN
        LISTDOC ( P ) ;
        IF NOT PRINTSW THEN NEXT_SCREEN ;
        P := P@.NEXT ;
       END ;
      END
   ELSE
    IF CODE = NAME1 THEN
     BEGIN
      SEARCHDOC ( W , FOUND , P ) ;
      IF NOT FOUND THEN DOCNOTFOUND ( W )
                   ELSE LISTDOC ( P ) ;
     END
```

```
    ELSE DISP28NL(ERRORMES);
         END;
END;


PROCEDURE DOCUMENTS;
 (* PRINT ALL DOCUMENT NAMES IN THE SYSTEM *)
VAR
  CH : CHAR ;
  P : DOCPTR ;
  NDOC, NCOL : SHORTINT ;
  Q : COLPTR ;
BEGIN
  NDOC := 0 ;
  IF PRINTSW THEN
   BEGIN
    HEADER ; BLANKS ( 44 ) ;
    WRITE28('DOCUMENTS IN THE SYSTEM     ');
    NEW_LINE ; NEW_LINE ;
   END ;
  FOR CH := 'A' TO 'Z' DO
  BEGIN
    P := DOCTABLE [ CH ] ;
    WHILE P <> NIL DO
     BEGIN
       NDOC := NDOC + 1 ;
       NCOL := 0 ;
       Q := P@.COLU ;
       WHILE Q<>NIL DO
        BEGIN
         NCOL := NCOL + 1 ;
         Q := Q@.NEXT ;
        END ;
       IF PRINTSW
        THEN
         BEGIN
          WRITEINT ( NDOC ) ; WRITE2 ('. ') ;
          WRITE28(P@.NAME); WRITE8('  WITH  ');
          WRITEINT ( NCOL ) ; WRITE8(' COLUMNS');
          NEW_LINE ;
         END
        ELSE
         BEGIN
          DISPINT ( NDOC ) ; DISPLAY ('.'); DISPLAY (' ');
          DISP28(P@.NAME); DISP8('  WITH  ');
          DISPINT ( NCOL ) ; DISP8(' COLUMNS');
          DISPLAY ( NL ) ;
         END;
       P := P@.NEXT ;
     END;
   END;
  IF NDOC = 0 THEN
                BEGIN
                  DISP8('NONE     '); DISPLAY(NL)
                  END ;
END;
```

```
PROCEDURE COLUMNS ;
  (* PRINT ALL COLUMN NAMES IN THE SYSTEM *)
VAR
  CH : CHAR ;
   P:COLTBPTR ;
   NLINES, NCOL : SHORTINT ;
BEGIN
  IF PRINTSW THEN
    BEGIN
     HEADER ; BLANKS ( 44 ) ;
     WRITE28(' COLUMNS IN THE SYSTEM      ');NEW_LINE;NEW_LINE;
    END;
  NLINES := 0 ;
  NCOL := 0 ;
  FOR CH := 'A' TO 'Z' DO
   BEGIN
    P := COLTABLE [ CH ] ;
    WHILE P <> NIL DO
     BEGIN
      NCOL := NCOL + 1 ;
      IF PRINTSW THEN BEGIN
                       WRITEINT ( NCOL ) ; WRITE2 ( '. ') ;
                       WRITE28(P@.NAME);
                       NEW_LINE ;
                      END
                 ELSE BEGIN
                       NLINES := NLINES + 1 ;
                       IF NLINES MOD SCREEN_LENGTH = 0 THEN NEXT_SCREEN ;
                       DISPINT(NCOL); DISPLAY('.'); DISPLAY(' ');
                       DISP28NL(P@.NAME) ;
                      END;
      P:=P@.NEXT ;
     END;
   END;
  IF NCOL = 0 THEN
    IF PRINTSW THEN BEGIN WRITE8('NONE    '); NEW_LINE END
               ELSE BEGIN DISP8('NONE    '); DISPLAY(NL) END;
END;


(*****************************************************************************)
(*                                                                           *)
(*              X R E F                 R O U T I N E S                      *)
(*                                                                           *)
(*****************************************************************************)


PROCEDURE XREFDC ( R : COLTBPTR ) ;
VAR
  Q : COLOCPTR ;
BEGIN
  IF PRINTSW
   THEN
    BEGIN
```

```
      WRITE28(R@.NAME);
      NEW_LINE ;
     END
    ELSE
     DISP28NL(R@.NAME);
   Q := R@.COLLIST ;
   WHILE Q <> NIL DO
    BEGIN
     IF PRINTSW
      THEN
       BEGIN
        BLANKS ( 28 ) ;
        WRITE28(Q@.PDOC@.NAME);
        NEW_LINE ;
       END
      ELSE
       BEGIN
        DISP8(BLANK8);
        DISP28NL(Q@.PDOC@.NAME);
       END;
     Q := Q@.NEXT ;
   END;
END;


PROCEDURE XREF;
VAR
  CH : CHAR ;
  R : COLTBPTR ;
  S : STATUS ;
  FOUND : BOOLEAN ;
  CODE : IDENT ;
  W : ALFA28 ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
  IF PRINTSW THEN
   BEGIN
    HEADER ; BLANKS ( 44 ) ;
    WRITE28('COLUMN  CROSS  REFERENCE    ');
    NEW_LINE; NEW_LINE ;
   END;
  CASE CODE OF
   SEMICOLON1 : BEGIN
                 FOR CH := 'A' TO 'Z' DO
                  BEGIN
                   R := COLTABLE [ CH ] ;
                   WHILE R <> NIL DO
                    BEGIN
                     XREFDC ( R ) ;
                     R := R@.NEXT ;
                     IF NOT PRINTSW THEN NEXT_SCREEN;
                    END;
                  END;
                END;
   NAME1       : BEGIN
```

```
                R := COLTABLE [ W [ 1 ] ] ;
                FOUND := FALSE ;
                WHILE (R <> NIL) AND (NOT FOUND) DO
                 IF R@.NAME = W THEN FOUND := TRUE
                               ELSE R := R@.NEXT ;
                IF FOUND THEN XREFDC ( R )
                         ELSE DISP28NL('COLUMN   NOT  FOUND         ');
              END;
   ELSE       :  DISP28NL('**** XREF - SYNTAX ERROR    ');
  END;
END;


(*****************************************************************************)
(*                                                                           *)
(*           F I N D                        R O U T I N E S                  *)
(*                                                                           *)
(*****************************************************************************)


FUNCTION MISSPELL ( W1 , W2 : ALFA28 ) : BOOLEAN ;
VAR
  LETTERS1, LETTERS2 : ARRAY [ 'A' .. 'Z' ] OF SHORTINT ;
  DIGITS1, DIGITS2 : ARRAY [ '0' .. '9' ] OF SHORTINT ;
  I , NDIF : SHORTINT ;
  CH : CHAR ;
BEGIN
  FOR CH := 'A' TO 'Z' DO
   BEGIN
    LETTERS1[CH]:= 0;
    LETTERS2[CH]:=0;
   END;
  FOR CH := '0' TO '9' DO
   BEGIN
    DIGITS1 [ CH ] := 0 ;
    DIGITS2 [ CH ] := 0 ;
   END;
  FOR I:= 1 TO NAMESIZE DO
   BEGIN
    CH := W1 [ I ] ;
     IF(CH<>' ')AND(CH<>' ')
       THEN
        IF CH IN DIGITS
         THEN DIGITS1[CH] := DIGITS1[CH] + 1
         ELSE LETTERS1[CH] := LETTERS1[CH] + 1 ;
     CH :=W2 [ I ] ;
     IF(CH<>'_')AND(CH<>' ')
      THEN
       IF CH IN DIGITS
        THEN DIGITS2[CH] := DIGITS2[CH] + 1
        ELSE LETTERS2[CH] := LETTERS2[CH] + 1 ;
   END ;
   NDIF := 0 ;
  FOR CH :='A' TO 'Z' DO
   NDIF := NDIF + ABS(LETTERS1[CH]-LETTERS2[CH]);
  FOR CH := '0' TO '9' DO
```

```
   NDIF := NDIF + ABS(DIGITS1[CH]-DIGITS2[CH]);
  MISSPELL := NDIF< 5;
END;


PROCEDURE FIND;
LABEL 1, 2 ;
VAR
  CODE, CODE1 : IDENT ;
  W : ALFA28 ;
  CH : CHAR ;
  P : DOCPTR ;
  R : COLTBPTR ;
  T : COLOCPTR ;
  FOUND : BOOLEAN ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W , CODE1 ) ;
  IF CODE1 = SIMILAR1
   THEN
    BEGIN
     NEXTSYM ( W , CODE ) ;
     IF CODE <> BLANK1 THEN GOTO 1 ;
     NEXTSYM ( W , CODE ) ;
     IF CODE <> TO1 THEN GOTO 1 ;
     NEXTSYM ( W , CODE ) ;
     IF CODE <> BLANK1 THEN GOTO 1 ;
     NEXTSYM ( W , CODE1 ) ;
    END;
  IF(CODE1<>DOCUMENT1)AND(CODE1<>COLUMN1) THEN GOTO 1 ;
  NEXTSYM ( W , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W , CODE ) ;
  IF CODE <> NAME1
   THEN
    GOTO 1
   ELSE
    BEGIN
     FOUND := FALSE ;
     IF CODE1 = DOCUMENT1
      THEN
       BEGIN
        FOR CH := 'A' TO 'Z' DO
         BEGIN
          P := DOCTABLE [ CH ] ;
          WHILE P <> NIL DO
           BEGIN
            IF MISSPELL ( P@.NAME , W )
             THEN
              BEGIN
               DISP28NL ( P@.NAME ) ;
               FOUND := TRUE ;
              END ;
            P := P@.NEXT ;
           END ;
```

```
          END ;
         IF NOT FOUND THEN DISP8 ('NONE    ');
         DISPLAY ( NL ) ;
        END
       ELSE
        BEGIN
         FOR CH := 'A' TO 'Z' DO
          BEGIN
           R := COLTABLE [ CH ] ;
           WHILE R <> NIL DO
            BEGIN
             IF MISSPELL ( R@.NAME , W )
              THEN
               BEGIN
                FOUND := TRUE ;
                T := R@.COLLIST ;
                 WHILE T <> NIL DO
                  BEGIN
                   DISP28BL ( R@.NAME ) ;
                   DISP8(' IN DOCU');
                   DISP8('MENT    ');
                   DISP28NL( T@.PDOC@.NAME ) ;
                   T := T@.NEXT ;
                  END;
               END;
             R := R@.NEXT ;
            END ;
          END ;
         IF NOT FOUND THEN DISP8('NONE    ') ;
         DISPLAY ( NL ) ;
        END
    END ;
  GOTO 2 ;
  1 : DISP28NL('**** FIND - SYNTAX ERROR    ');
  2 :
END;


(*************************************************************************)
(*                                                                       *)
(*                K  E  Y                    R O U T I N E S           *)
(*                                                                       *)
(*************************************************************************)


PROCEDURE KEYLIST ( R : COLPTR ) ;
VAR R1 : COLPTR ;
BEGIN
  R1 := R ;
  WHILE R1 <> NIL DO
   BEGIN
    DISP8(BLANK8);
    DISP28NL(R1@.PNAME@.NAME) ;
    R1@.KEYFD := NOTDECLARED ;
    R1 := R1@.NEXT ;
   END;
```

```
END;


PROCEDURE INKEYS ( R1 : COLPTR ) ;
VAR
  R : COLPTR ;
  S : STATUS ;
  W : ALFA28 ;
  CODE : IDENT ;
  ERRORSW, FOUND : BOOLEAN ;
BEGIN
  IF RETRIEVESW
   THEN
    GETCOMD1
   ELSE
    BEGIN
     DISP28NL('TYPE NAMES OF KEY COLUMNS   ') ;
     GETCOMD ( S ) ;
    END ;
  ERRORSW := FALSE ;
  REPEAT
   R := R1 ;
   NEXTSYM ( W , CODE ) ;
   IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
   CASE CODE OF
    SEMICOLON1 : ;
    NAME1      : BEGIN
                   FOUND := FALSE ;
                   WHILE(R <> NIL) AND (NOT FOUND) DO
                    IF R@.PNAME@.NAME = W THEN FOUND := TRUE
                                          ELSE R := R@.NEXT ;
                   IF FOUND THEN R@.KEYFD := DECLARED
                            ELSE BEGIN
                           DISP8('COLUMN  '); DISP28BL( W ) ;
                           DISP8('NOT FOUN'); DISPLAY('D');
                           DISPLAY(NL);
                          END;
                 END;
    ELSE : ERRORSW := TRUE ;
     END;
   UNTIL (ERRORSW) OR (CODE=SEMICOLON1);
   IF ERRORSW THEN DISP28NL('ERROR IN COLUMN LIST       ');
END;


PROCEDURE KEY;
VAR
  W : ALFA28 ;
  CODE : IDENT ;
  CH : CHAR ;
  P : DOCPTR ;
  FOUND : BOOLEAN ;
  R : COLPTR ;
BEGIN
  IF NOT RETRIEVESW THEN WAS_A_SV := FALSE ;
  NEXTSYM ( W , CODE ) ;
```

```
  IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
  IF CODE = SEMICOLON1
   THEN
    BEGIN
     FOR CH := 'A' TO 'Z' DO
      BEGIN
       P := DOCTABLE [ CH ] ;
       WHILE P <> NIL DO
        BEGIN
         DISP28NL ( P@.NAME ) ;  R := P@.COLU ;
         IF R <> NIL THEN
          BEGIN
           KEYLIST ( R ) ;
           INKEYS ( R ) ;
           P@.KEYFD := DECLARED ;
          END;
         P := P@.NEXT ;
        END ;
     END
  END
    ELSE
     IF CODE = NAME1
      THEN
       BEGIN
        SEARCHDOC ( W , FOUND , P ) ;
        IF FOUND
         THEN
          BEGIN
           IF NOT RETRIEVESW THEN DISP28NL ( P@.NAME ) ;
           R := P@.COLU ;
           IF R <> NIL THEN
            BEGIN
             IF NOT RETRIEVESW THEN KEYLIST ( R ) ;
             INKEYS ( R ) ;
             P@.KEYFD := DECLARED ;
            END
        END
          ELSE DOCNOTFOUND ( W ) ;
      END
     ELSE
       DISP28NL('**** KEY - SYNTAX ERROR     ');
END;


(*************************************************************************)
(*                                                                       *)
(*        P R E P A R E              R O U T I N E S                     *)
(*                                                                       *)
(*************************************************************************)


PROCEDURE PRCL ( D: DECLARATION; R1: COLPTR; VAR NCOL_PR : SHORTINT);
  (* WRITE ALL COLUMNS WHOSE KEY IS D FOR A GIVEN DOCUMENT *)
CONST NCOL_PR_LINE = 17 ;
VAR
  R : COLPTR ;
```

```
  FIRST : BOOLEAN ;
BEGIN
  R := R1 ;
  FIRST := TRUE ;
  WHILE R <> NIL DO
   BEGIN
    IF R@.KEYFD=D THEN
     BEGIN
      IF NCOL_PR > NCOL_PR_LINE THEN
       BEGIN
        WRITE ( NL ); BLANKS ( 3 ) ;
        NCOL_PR := 0;
       END;
      IF NOT FIRST THEN WRITE(',');
      FIRST := FALSE ;  NCOL_PR := NCOL_PR + 1 ;
      WRITE3(R@.PNAME@.ABV);
     END;
    R := R@.NEXT ;
   END
END;


PROCEDURE PREPARE ;
VAR
  CH : CHAR ;
  P : COLTBPTR ;
  N : ARRAY [ 0 .. 9 ] OF CHAR ;
  I2 , I3 , NCOL_PR : SHORTINT ;
  W : ALFA3 ;
  Q : DOCPTR ;
  R : COLPTR ;
  BLK1, BLK2, BLK3 : SVC7_BLOCK ;
BEGIN
  N[0] := '0' ; N[1] := '1' ; N[2] := '2' ; N[3] := '3' ; N[4] := '4' ;
  N[5] := '5' ; N[6] := '6' ; N[7] := '7' ; N[8] := '8' ; N[9] := '9' ;
  (* PREPARE ABBREVIATED COLUMN NAMES *)
  HEADER ; BLANKS ( 44 ) ;
  WRITE28('  DICTIONARY - BERNSTEIN"S A');
  WRITE8('LGORYTHM'); NEW_LINE; NEW_LINE ;
  FOR CH := 'A' TO 'Z' DO
   BEGIN
    P := COLTABLE [ CH ] ;
    I2 := 0 ; I3 := 0 ; W [ 1 ] := CH ;
    WHILE P <> NIL DO
     BEGIN
      W [ 2 ] := N [ I2 ] ;  W [ 3 ] := N [ I3 ] ;
      P@.ABV := W ;
      WRITE28(P@.NAME); WRITE3(W); WRITE ( NL ) ;
      I3 := I3 + 1 ;
      IF I3 > 9 THEN BEGIN
                      I3 := 0 ; I2 := I2 + 1 ;
                     END;
      P := P@.NEXT ;
     END;
   END;
  WITH BLK1 DO
```

```
    BEGIN
     SVC7_CMD := SVC7_CLOSE ;
     SVC7_MOD := SVC7_BUF_DEFAULT ;
     SVC7_LU  := 2 ;
    END;
   WITH BLK2 DO
    BEGIN
     SVC7_CMD := SVC7_ASSIGN ;
     SVC7_MOD := SVC7_AP_ERW ;
     SVC7_LU  := 2 ;
     SVC7_KEYS:= 0 ;
     SVC7_RECLEN := 512 ;
     SVC7_FD.VOLN := 'USR6' ;
     SVC7_FD.FN   := 'BERNDAT ' ;
     SVC7_FD.EXTN := 'LIS' ;
     SVC7_FD.ACCT := 'P' ;
    END;
   WITH BLK3 DO
    BEGIN
     SVC7_CMD := SVC7_ASSIGN ;
     SVC7_MOD := SVC7_BUF_DEFAULT ;
     SVC7_LU  := 2 ;
     SVC7_FD.VOLN := 'PR  ' ;
     SVC7_FD.FN   := '        ' ;
     SVC7_FD.EXTN := '   ' ;
     SVC7_FD.ACCT := ' ' ;
    END ;
   SVC7 ( BLK1 ) ;
   SVC7 ( BLK2 ) ;
   RESET ( 2 ) ;
   FOR CH := 'A' TO 'Z' DO
    BEGIN
     Q := DOCTABLE [ CH ] ;
     WHILE Q <> NIL DO
      BEGIN
       NCOL_PR := 0 ;
       IF Q@.KEYFD = DECLARED THEN
        BEGIN
         R := Q@.COLU ;
         IF R <> NIL THEN
          BEGIN
           PRCL ( DECLARED , R , NCOL_PR ) ;
           WRITE ('>') ;
           PRCL ( NOTDECLARED , R , NCOL_PR ) ;
           WRITE(';'); WRITE(NL);
          END;
        END;
       Q := Q@.NEXT ;
      END;
     END;
   WRITE3('END'); WRITE('.'); WRITE(EM);
   SVC7 ( BLK1 ) ;
   SVC7 ( BLK3 ) ;
   RESET ( 2 ) ;
END;
```

```
(*************************************************************************)
(*                                                                       *)
(*       A D D / D E L E T E          R O U T I N E S                    *)
(*                                                                       *)
(*************************************************************************)


PROCEDURE COLLECTAT ( VAR P1 : ATRPTR ; FIRST : ATTRIBPTR ) ;
LABEL 1 ;
VAR P : ATTRIBPTR ;
    Q : COMPTR ;
    W1 , W2 : ALFA28 ;
    ENDREP , ERROR : BOOLEAN ;
    R , R1 : ATRPTR ;
    CODE : IDENT ;
BEGIN
  R1 := NIL ;
  ERROR := FALSE ;
  NEXTSYM ( W1 , CODE ) ;
  REPEAT
   P := FIRST ;
   IF CODE = BLANK1 THEN NEXTSYM ( W1 , CODE ) ;
   IF CODE = RIGHTPAR1 THEN GOTO 1 ;
   IF CODE <> NAME1 THEN
      BEGIN
       ERROR := TRUE ;
       GOTO 1 ;
      END;
   NEXTSYM ( W2 , CODE ) ;
   IF CODE = BLANK1 THEN NEXTSYM ( W2 , CODE ) ;
   IF CODE <> PERIOD1 THEN
      BEGIN
       ERROR := TRUE ;
       GOTO 1 ;
      END ;
   NEXTSYM ( W2 , CODE ) ;
   IF CODE = BLANK1 THEN NEXTSYM ( W2 , CODE ) ;
   IF CODE <> NAME1 THEN
      BEGIN
       ERROR := TRUE ;
       GOTO 1 ;
      END ;
    ENDREP := FALSE ;
    REPEAT
      IF P <> NIL THEN
         IF P@.NAME=W1 THEN ENDREP := TRUE
                       ELSE P:=P@.NEXT;
    UNTIL (P=NIL) OR ENDREP ;
    IF NOT ENDREP THEN
      BEGIN
        DISP8('ATTRIBUT');
        DISPLAY('E');DISPLAY(' ');DISP28BL(W1);DISP8('NOT FOUN');
        DISPLAY('D'); DISPLAY(NL);
      END
     ELSE
```

```
      BEGIN
        Q:=P@.COM ;
        ENDREP := FALSE ;
        REPEAT IF Q<>NIL THEN
          IF Q@.NAME = W2 THEN ENDREP := TRUE
                         ELSE Q := Q@.NEXT ;
        UNTIL (Q=NIL) OR ENDREP ;
        IF NOT ENDREP THEN
         BEGIN
           DISP8('COMPONEN');DISPLAY('T');DISPLAY(' ');DISP28BL(W2);
           DISP8('NOT FOUN'); DISPLAY('D'); DISPLAY(NL);
         END
        ELSE
         BEGIN
           NEW ( R ) ;
           R@.COMPT := Q ;
           R@.NEXT := NIL ;
           IF R1=NIL THEN P1 := R
                     ELSE R1@.NEXT := R ;
              R1 := R ;
            END;
         END;
        NEXTSYM( W1 , CODE ) ;
        IF CODE = BLANK1 THEN NEXTSYM ( W1 , CODE ) ;
      1:
     UNTIL(CODE=RIGHTPAR1)OR(CODE=SEMICOLON1)OR ERROR ;
    IF ERROR THEN
       DISP28NL('ATTRIBUTE LIST ERROR        ');
END;


PROCEDURE SEARCHDOC ;
 (* SEARCH FOR DOCUMENT W, FOUND WILL BE TRUE IF W WAS FOUND,
    P WILL POINT TO THAT DOCUMENT *)
BEGIN
  FOUND := FALSE ;
  P1 := DOCTABLE [ W [1] ] ;
  WHILE (P1<>NIL)AND(NOT FOUND) DO
   IF P1@.NAME=W THEN FOUND := TRUE
                 ELSE P1 := P1@.NEXT ;
END;


PROCEDURE DELTCOL(W:ALFA28;VAR P1:DOCPTR;VAR S:STATUS;VAR R:ATRPTR);
 (* DELETE COLUMN W FROM DOCUMENT POINTED BY P1. S IS THE STATUS
AFTER OPERATION, R IS A POINTER TO THE LIST OF ATTRIBUTES *)
VAR
  T,T1:COLPTR ;
  U,U1:COLTBPTR ;
  V,V1:COLOCPTR ;
FOUND : BOOLEAN ;
BEGIN
  S := WASOK ;
  T := P1@.COLU ; FOUND := FALSE ;
  T1 := T ;
  WHILE(T<>NIL)AND(NOT FOUND) DO
```

```
   IF T@.PNAME@.NAME=W THEN FOUND := TRUE
     ELSE BEGIN
           T1 := T ;
           T := T@.NEXT ;
          END;
  IF NOT FOUND THEN
     BEGIN
       COLNOTFOUND(W,P1@.NAME);
       S := WASNOTOK ;
     END
    ELSE
     BEGIN
      R:= T@.ATRP ;
      IF T=P1@.COLU THEN P1@.COLU:=T@.NEXT
                    ELSE T1@.NEXT := T@.NEXT ;
       (* NOW TAKE CARE OF OCCURANCE *)
      U := COLTABLE[W[1]]; U1 := U ;
      FOUND := FALSE;
      WHILE (U<>NIL)AND(NOT FOUND) DO
       IF U@.NAME=W THEN FOUND := TRUE
                    ELSE BEGIN U1 := U;
                               U:=U@.NEXT
                         END;
      (* NOW U POINTS TO REQUIRED COLUMN NAME IN COLTABLE *)
      V:= U@.COLLIST; V1:=V;FOUND:=FALSE;
      WHILE(V<>NIL)AND(NOT FOUND) DO
        IF V@.PDOC = P1 THEN FOUND := TRUE
              ELSE BEGIN V1 := V ; V := V@.NEXT ; END;
       IF V=V1 THEN U@.COLLIST:=V@.NEXT
               ELSE V1@.NEXT:=V@.NEXT ;
       IF U@.COLLIST = NIL   (* NO OCCURANCES *)
         THEN IF U = U1 THEN COLTABLE[W[1]]:=U@.NEXT
                        ELSE U1@.NEXT := U@.NEXT ;
      END
END;


PROCEDURE ADCOL ( VAR P:DOCPTR; W:ALFA28; VAR RCL:COLPTR);
  (* P POINTS TO DOCUMENT THAT WILL RECEIVE THE NEW COLUMN ,
     W IS THE NAME OF THE COLUMN, R IS A POINTER TO THE COLREC
     RECORD (LINKED TO THE DOCUMENT), WILL BE USED TO APPEND
     THE LIST OF ATTRIBUTES *)
LABEL 1 ;
VAR
  CH : CHAR ;
  R, R1 : COLTBPTR ;
  FOUND : BOOLEAN ;
  Q, Q1 : COLOCPTR ;
  T, T1 : COLPTR ;
BEGIN
  CH := W [ 1 ] ;
  IF COLTABLE [ CH ] = NIL
    THEN  BEGIN
            NEW ( R ) ;
            R@.NAME := W ;
            COLTABLE [ CH ] := R ;
```

```
            R@.COLLIST := NIL ;
            R@.NEXT := NIL ;
        END
  ELSE  BEGIN
            R:= COLTABLE [ CH ] ;
            R1 := R ; FOUND := FALSE ;
            WHILE ( R<>NIL ) AND ( NOT FOUND ) DO
             IF R@.NAME=W THEN FOUND := TRUE
                          ELSE BEGIN
                               R1 := R ;
                               R := R@.NEXT ;
                               END;
            IF R=NIL THEN BEGIN
                           NEW(R); R@.NAME:=W ;
                           R1@.NEXT := R ;
                           R@.NEXT := NIL ;
                           R@.COLLIST := NIL ;
                          END;
        END ;
(*  R NOW POINTS TO REQUIREDRECORD IN COLTAB *)
(*  LOOK FOR SAME OCCURANCE OF THE COLUMN *)
Q := R@.COLLIST ;
IF Q = NIL THEN BEGIN
                 NEW ( Q ) ; R@.COLLIST := Q ;
                 Q@.NEXT := NIL ;
                 Q@.PDOC := P ;
                 Q@.PCOL := NIL
                END
           ELSE BEGIN
   FOUND := FALSE ;
   WHILE ( Q<>NIL ) AND ( NOT FOUND ) DO
    IF Q@.PDOC = P THEN FOUND := TRUE
                   ELSE BEGIN
                         Q1 := Q ;
                         Q := Q@.NEXT ;
                        END;
    IF FOUND THEN BEGIN
                   DISP8('ERROR - ');
                   DISP8('COLUMN  '); DISP28BL ( W ) ;
                   DISP28NL(' ALREADY EXISTS              ');
                   GOTO 1 ;
                  END;
    NEW ( Q ) ;   (* ESTABLISH NEW OCCURANCE *)
    Q1@.NEXT := Q ;
    Q@.PDOC := P ;
    Q@.NEXT := NIL ;
    Q@.PCOL := NIL ;
   END;
  (* NOW Q POINTS TO THE COLUMN OCCURANCE, A COLREC HAS TO BE
     ADDED NOW TO THE DOCUMENT  *)
  NEW(T) ; RCL := T ;
  IF P@.COLU = NIL THEN P@.COLU := T
                   ELSE BEGIN T1 := P@.COLU;
                         WHILE T1@.NEXT <> NIL DO
                                          T1 := T1@.NEXT ;
                         T1@.NEXT := T ;
```

```
                              END;
    T@.NEXT := NIL ;
    T@.KEYFD := NOTDECLARED ;
    T@.ATRP := NIL ;
    T@.PNAME := R ;
    Q@.PCOL := T ;
   1:
END;


PROCEDURE SEARCHATR (FIRST : ATTRIBPTR ; W1 , W2 : ALFA28 ;
                     VAR FOUND : BOOLEAN ; VAR R : COMPTR ) ;
(* CHECK IF COMPONENT W2 EXISTS IN ATTRIBUTE W1 *)
VAR
  S : ATTRIBPTR ;
BEGIN
  FOUND := FALSE ;
  S := FIRST ;
  WHILE (S<>NIL) AND (NOT FOUND) DO
   IF S@.NAME = W1 THEN FOUND := TRUE
                   ELSE S := S@.NEXT ;
  IF FOUND
   THEN
    BEGIN
     R := S@.COM ;
     FOUND := FALSE ;
     WHILE (R<>NIL)AND(NOT FOUND) DO
      IF R@.NAME = W2 THEN FOUND := TRUE
                      ELSE R := R@.NEXT ;
    END;
END;


PROCEDURE ADDATR ;
(* ADD OR DELETE ATTRIBUTE TO/FROM A DOCUMENT OR A COLUMN *)
LABEL 1;
CONST
  MESSAGE1 = 'ATTRIB/COMPON NOT FOUND      ' ;
  MESSAGE2 = 'ATTRIBUTE ALREADY EXISTS     ' ;
  MESSAGE3 = 'ATTRIBUTE DOES NOT EXIST     ' ;
  MESSAGE4 = '**** ADD - SYNTAX ERROR      ' ;
  MESSAGE5 = '**** DELETE - SYNTAX ERROR   ' ;
VAR
  CODE, CODE1   : IDENT ;
  W1, W2, W3, W4 : ALFA28 ;
  FOUND : BOOLEAN ;
  P : DOCPTR ;
  T,T1 : ATRPTR ;
  R : COMPTR ;
  Q : COLPTR ;
BEGIN
  NEXTSYM ( W1 , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1;
  NEXTSYM ( W1 , CODE ) ;
  IF CODE <> NAME1 THEN GOTO 1 ;
  NEXTSYM ( W2 , CODE ) ;
```

```
IF CODE = BLANK1 THEN NEXTSYM ( W2 , CODE ) ;
IF CODE <> PERIOD1 THEN GOTO 1 ;
NEXTSYM ( W2 , CODE ) ;
IF CODE = BLANK1 THEN NEXTSYM ( W2 , CODE ) ;
IF CODE <> NAME1 THEN GOTO 1 ;
(* W1 . W2 WERE FOUND *)
NEXTSYM ( W3 , CODE ) ;
IF CODE <> BLANK1 THEN GOTO 1 ;
NEXTSYM ( W3 , CODE ) ;
IF CODE <> TO1 THEN GOTO 1 ;
NEXTSYM ( W3 , CODE ) ;
IF CODE <> BLANK1 THEN GOTO 1 ;
NEXTSYM ( W3 , CODE1 ) ;
IF(CODE1<>DOCUMENT1)AND(CODE1<>COLUMN1)THEN GOTO 1 ;
NEXTSYM ( W3 , CODE ) ;
IF CODE <> BLANK1 THEN GOTO 1 ;
NEXTSYM ( W3 , CODE ) ;
IF CODE <> NAME1 THEN GOTO 1 ;
IF CODE1 = DOCUMENT1
 THEN
  BEGIN
   SEARCHDOC ( W3 , FOUND , P ) ;
   IF NOT FOUND
    THEN DOCNOTFOUND ( W3 )
    ELSE
     BEGIN
      SEARCHATR ( DOCATFIRST , W1 , W2 , FOUND , R ) ;
      IF NOT FOUND
       THEN DISP28NL ( MESSAGE1 )
       ELSE
        BEGIN
         T := P@.ATRP ; T1 := T ;
         FOUND := FALSE ;
         WHILE (T <> NIL) AND (NOT FOUND) DO
          IF T@.COMPT = R THEN FOUND := TRUE
                          ELSE BEGIN T1 := T; T := T@.NEXT ;
                               END;
         IF FOUND
          THEN IF DELTSW THEN
            IF T1 = P@.ATRP THEN P@.ATRP := T@.NEXT
                            ELSE T1@.NEXT := T@.NEXT
          ELSE DISP28NL ( MESSAGE2 )
         ELSE
          IF DELTSW THEN DISP28NL ( MESSAGE3 )
           ELSE
           BEGIN
            NEW ( T ) ;
            T@.NEXT := P@.ATRP ;
            P@.ATRP := T ;
            T@.COMPT := R ;
           END
        END
     END
  END
 ELSE
  BEGIN
```

```
NEXTSYM ( W4 , CODE ) ;
IF CODE <> BLANK1 THEN GOTO 1 ;
NEXTSYM ( W4 , CODE ) ;
IF CODE <> IN1 THEN GOTO 1 ;
NEXTSYM ( W4 , CODE ) ;
IF CODE <> BLANK1 THEN GOTO 1;
NEXTSYM ( W4 , CODE ) ;
IF  CODE = DOCUMENT1
 THEN
  BEGIN
   NEXTSYM ( W4 , CODE ) ;
   IF CODE <> BLANK1 THEN GOTO 1 ;
   NEXTSYM ( W4 , CODE ) ;
  END ;
IF CODE <> NAME1
 THEN
  BEGIN 1:
     IF DELTSW THEN DISP28NL ( MESSAGE5 )
               ELSE DISP28NL ( MESSAGE4 )
  END
 ELSE
  BEGIN
   SEARCHDOC ( W4 , FOUND , P ) ;
   IF NOT FOUND
    THEN DOCNOTFOUND ( W4 )
    ELSE
     BEGIN
      Q := P@.COLU ;
      FOUND := FALSE ;
      WHILE (Q<>NIL)AND(NOT FOUND) DO
       IF Q@.PNAME@.NAME = W3 THEN FOUND := TRUE
                              ELSE Q := Q@.NEXT ;
      IF NOT FOUND
       THEN COLNOTFOUND ( W3 , W4 )
       ELSE
        BEGIN
         SEARCHATR ( COLATFIRST , W1 , W2 , FOUND , R ) ;
         IF NOT FOUND
          THEN DISP28NL ( MESSAGE1 )
          ELSE
           BEGIN
            T := Q@.ATRP ; T1 := T ;
            FOUND := FALSE ;
            WHILE (T <> NIL) AND (NOT FOUND) DO
             IF T@.COMPT = R THEN FOUND := TRUE
                             ELSE BEGIN T1:=T;T:=T@.NEXT END;
             IF FOUND
              THEN
              IF DELTSW
               THEN IF T1=Q@.ATRP THEN Q@.ATRP := T@.NEXT
                                  ELSE T1@.NEXT := T@.NEXT
               ELSE DISP28NL ( MESSAGE2 )
              ELSE IF DELTSW THEN DISP28NL ( MESSAGE3 ) ELSE
               BEGIN
                NEW ( T ) ;
                T@.NEXT := Q@.ATRP ;
```

```
                    Q@.ATRP := T ;
                    T@.COMPT := R ;
                  END ;
              END
          END
      END
    END
END;


PROCEDURE ADDDOC ;
LABEL 1,2 ;
VAR
  DOCNAME, W : ALFA28 ;
  S : STATUS ;
  FOUND : BOOLEAN ;
  CODE : IDENT ;
  P, P1, Q : DOCPTR ;
  RCL : COLPTR ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( DOCNAME , CODE ) ;
  IF CODE <> NAME1 THEN GOTO 1 ;
  FOUND := FALSE ;
  P := DOCTABLE [ DOCNAME[1] ] ;
  P1 := P ;
  WHILE (P<>NIL)AND(NOT FOUND) DO
   IF P@.NAME = DOCNAME THEN FOUND := TRUE
                        ELSE BEGIN P1 := P ;
                                   P := P@.NEXT ;
                             END;
  IF FOUND THEN
    BEGIN
      DISP28('DOCUMENT ALREADY DECLARED   ');
      DISPLAY ( NL ) ;
      GOTO 2 ;
    END;
  NEW ( Q ) ;
  IF P = P1 THEN DOCTABLE [ DOCNAME[1] ] := Q
            ELSE P1@.NEXT := Q ;
  WITH Q@ DO
    BEGIN
      NAME := DOCNAME ;
      NEXT := NIL ;
      ATRP := NIL ;
      COLU := NIL ;
      KEYFD := NOTDECLARED ;
    END;
  NEXTSYM ( W , CODE ) ;
  IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
  IF CODE = LEFTPAR1
     THEN COLLECTAT ( Q@.ATRP , DOCATFIRST )
     ELSE IF CODE <> SEMICOLON1
       THEN BEGIN
```

```
            1: DISP28('**** ADD - SYNTAX ERROR     ');
               DISPLAY(NL);
               GOTO 2;
            END;
  IF NOT RETRIEVESW THEN
  BEGIN
  DISP28('INSERT COLUMN NAMES FOR DOCU');
  DISP8('MENT :  '); DISP28(DOCNAME); DISPLAY(NL);
  END;
  IF RETRIEVESW THEN GETCOMD1 ELSE GETCOMD ( S ) ;
  NEXTSYM ( W , CODE ) ;
  IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
  WHILE CODE <> SEMICOLON1 DO
   BEGIN
      IF CODE<>NAME1 THEN BEGIN
                          DISP28('WRONG COLUMN IN COLUMN LIST ');
                          DISPLAY(NL); GOTO 2;
                          END;
     ADCOL(Q,W,RCL);
      NEXTSYM( W , CODE ) ;
      IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
      IF CODE = LEFTPAR1 THEN BEGIN COLLECTAT(RCL@.ATRP,COLATFIRST);
      NEXTSYM(W,CODE);
      IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ; END;
    END;
 2:
END;


PROCEDURE ADDCOL;
LABEL 1;
VAR
  W1,W2:ALFA28;
  CODE : IDENT ;
  P1 : ATRPTR ;
  P : DOCPTR ;
  FOUND : BOOLEAN ;
  RCL : COLPTR ;
BEGIN
  P1 := NIL ;
  NEXTSYM ( W1 , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W1 , CODE ) ;
  IF CODE <> NAME1 THEN GOTO 1 ;
  NEXTSYM ( W2 , CODE ) ;
  IF CODE = BLANK1 THEN NEXTSYM ( W2 , CODE ) ;
  IF CODE = LEFTPAR1 THEN
     BEGIN COLLECTAT ( P1 , COLATFIRST ) ;
           NEXTSYM ( W2 , CODE ) ;
           IF CODE = BLANK1 THEN NEXTSYM ( W2 , CODE ) ;
     END;
  IF CODE <> TO1 THEN GOTO 1 ;
  NEXTSYM ( W2 , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W2 , CODE ) ;
  IF CODE = DOCUMENT1 THEN
```

```
      BEGIN NEXTSYM ( W2 , CODE ) ;
      IF CODE <> BLANK1 THEN GOTO 1 ;
      END ;
   NEXTSYM ( W2 , CODE ) ;
   IF CODE <> NAME1 THEN
      BEGIN 1:
            DISP28NL ( '**** ADD - SYNTAX ERROR     ');
      END
     ELSE
      BEGIN
           SEARCHDOC ( W2 , FOUND , P ) ;
           IF NOT FOUND
            THEN DISP28NL('DOCUMENT NOT FOUND          ')
            ELSE
             BEGIN
              ADCOL ( P , W1 , RCL ) ;
              RCL@.ATRP := P1
             END;
      END
END;


PROCEDURE ADD;
LABEL 1 ;
VAR
  W : ALFA28 ;
  CODE : IDENT ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W , CODE ) ;
  CASE CODE OF
    DOCUMENT1   : ADDDOC ;
    COLUMN1     : ADDCOL ;
    ATTRIBUTE1  : ADDATR ;
    ELSE        : BEGIN
                  1: DISP28('**** ADD - SYNTAX ERROR     ');
                     DISPLAY ( NL ) ;
                  END;
   END ;
  IF NOT RETRIEVESW THEN WAS_A_SV := FALSE ;
END;


PROCEDURE RENAME;
 (* USED TO RENAME A DOCUMENT OR A COLUMN INSIDE A DOCUMENT *)
LABEL 1,2 ;
VAR
  W1, W2, W3 : ALFA28 ;
  CODE1, CODE : IDENT ;
  P1 : DOCPTR ;
  RCL, P2 : COLPTR ;
  FOUND : BOOLEAN ;
  S : STATUS ;
  R : ATRPTR ;
BEGIN
```

```
  NEXTSYM(W1,CODE); IF CODE<>BLANK1 THEN GOTO 1;
  NEXTSYM(W1,CODE1); IF(CODE1<>DOCUMENT1)AND(CODE1<>COLUMN1)THEN GOTO 1 ;
  NEXTSYM(W1,CODE); IF CODE<>BLANK1 THEN GOTO 1;
  NEXTSYM(W1,CODE); IF CODE<>NAME1 THEN GOTO 1;
  NEXTSYM(W2,CODE); IF CODE<>BLANK1 THEN GOTO 1 ;
  NEXTSYM(W2,CODE); IF CODE<>AS1 THEN GOTO 1 ;
  NEXTSYM(W2,CODE); IF CODE<>BLANK1 THEN GOTO 1;
  NEXTSYM(W2,CODE); IF CODE<>NAME1 THEN GOTO 1;
  SEARCHDOC ( W2 , FOUND , P1 ) ;
  IF FOUND THEN BEGIN
                DISP8('DOCUMENT');DISPLAY(' '); DISP28BL(W2);
                DISP28NL('ALREADY EXISTS              '); GOTO 2;
                END;
  IF W1=W2 THEN BEGIN
                 DISP8('REDUNDAN');DISPLAY('T');
                 DISPLAY(NL); GOTO 2 ;
                END;
  NEXTSYM(W3,CODE); IF CODE=BLANK1 THEN NEXTSYM(W3,CODE);
  IF(CODE=SEMICOLON1)AND(CODE1=DOCUMENT1)
   THEN
    BEGIN
     SEARCHDOC ( W1 , FOUND , P1 ) ;
     IF FOUND THEN P1@.NAME := W2
              ELSE BEGIN DOCNOTFOUND ( W1 ); GOTO 2 END;
     GOTO 2 ;
    END
   ELSE IF (CODE<>SEMICOLON1)AND(CODE1=DOCUMENT1) THEN GOTO 1 ;
   (* NOW CHECK FOR COLUMN RENAME *)
   IF CODE<>IN1 THEN GOTO 1 ;
  NEXTSYM(W3,CODE); IF CODE<>BLANK1 THEN GOTO 1;
  NEXTSYM(W3,CODE);
  IF CODE=DOCUMENT1 THEN BEGIN
                          NEXTSYM(W3,CODE); IF CODE<>BLANK1 THEN GOTO 1;
                          NEXTSYM(W3,CODE);
                         END;
  IF CODE<>NAME1 THEN GOTO 1 ;
  (* RENAME COLUMN W1 AS W2 IN DOCUMENT W3 *)
  IF CODE1 = COLUMN1 THEN
   BEGIN
    SEARCHDOC(W3,FOUND,P1);
    IF NOT FOUND THEN
     BEGIN DOCNOTFOUND(W3);
           GOTO 2;
     END;
    DELTCOL(W1,P1,S,R) ;
    IF S = WASNOTOK THEN GOTO 2 ;
    ADCOL( P1 , W2 , RCL ) ;
    RCL@.ATRP := R ;
   END;
   GOTO 2 ;
   1: DISP28NL('**** RENAME - SYNTAX ERROR  ');
   2: WAS_A_SV := FALSE ;
END;


PROCEDURE DELETE;
```

```
LABEL 1;
VAR
  W1,W2 : ALFA28 ;
  CODE, CODE1 : IDENT ;
  P, P1 : DOCPTR ;
  FOUND : BOOLEAN ;
  R : ATRPTR ;
  S : STATUS ;
  Q : COLPTR ;
BEGIN
  WAS_A_SV := FALSE ;
  NEXTSYM ( W1 , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W1 , CODE1 ) ;
IF(CODE1<>COLUMN1)AND(CODE1<>DOCUMENT1)AND(CODE1<>ATTRIBUTE1)THEN GOTO 1 ;
  IF CODE1=ATTRIBUTE1 THEN BEGIN
                            DELTSW := TRUE ;
                            ADDATR ;
                            DELTSW := FALSE ;
                           END
  ELSE
  BEGIN
  NEXTSYM ( W1 , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W1 , CODE ) ;
  IF CODE <> NAME1 THEN GOTO 1 ;
  IF CODE1 = DOCUMENT1 THEN
   BEGIN
     (* DELETE DOCUMENT *) ;
     P := DOCTABLE [ W1 [1] ] ;
     P1 := P ;
     FOUND := FALSE ;
     WHILE (P <> NIL) AND (NOT FOUND) DO
      IF P@.NAME=W1 THEN FOUND := TRUE
                    ELSE BEGIN
                           P1 := P ;
                           P := P@.NEXT ;
                          END;
     IF NOT FOUND THEN DOCNOTFOUND ( W1 )
       ELSE BEGIN IF P1=P THEN DOCTABLE [ W1[1] ] := P@.NEXT
                  ELSE P1@.NEXT := P@.NEXT ;
            Q := P@.COLU ;
            WHILE Q<>NIL DO BEGIN
                              DELTCOL(Q@.PNAME@.NAME,P,S,R);
                              Q := Q@.NEXT ;
                            END;
            END;
   END
   ELSE
   BEGIN
    (* DELETE COLUMN *)
    NEXTSYM ( W2 , CODE ) ;
    IF CODE <> BLANK1 THEN GOTO 1 ;
    NEXTSYM ( W2 , CODE ) ;
    IF CODE <> TO1 THEN GOTO 1 ;
    NEXTSYM ( W2 , CODE ) ;
```

```
    IF CODE <> BLANK1 THEN GOTO 1 ;
    NEXTSYM ( W2 , CODE ) ;
    IF CODE = DOCUMENT1 THEN
     BEGIN
      NEXTSYM ( W2 , CODE ) ;
      IF CODE <> BLANK1 THEN GOTO 1 ;
      NEXTSYM ( W2 , CODE ) ;
     END ;
    IF CODE <> NAME1 THEN
       BEGIN 1:
             DISP28NL('**** DELETE - SYNTAX ERROR  ');
       END
     ELSE
       BEGIN
          SEARCHDOC ( W2 , FOUND , P ) ;
          IF NOT FOUND THEN DOCNOTFOUND ( W2 )
                ELSE  DELTCOL (W1,P,S,R);
       END ;
     END
 END;
END;


(*****************************************************************************)
(*                                                                           *)
(*          D E F I N E                    R O U T I N E S                   *)
(*                                                                           *)
(*****************************************************************************)


PROCEDURE DEFINE;
LABEL 1 ;
VAR
  KIND, CODE : IDENT;
  ATNAME,WD : ALFA28;
  FIRST,P,Q: COMPTR ;
  ENDREP : BOOLEAN ;
  Q1,P1 : ATTRIBPTR ;
BEGIN
  NEXTSYM ( WD , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( WD , CODE ) ;
  IF (CODE<>DOCUMENT1)AND(CODE<>COLUMN1)THEN GOTO 1;
  KIND := CODE ;
  NEXTSYM ( WD , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1;
  NEXTSYM ( WD , CODE ) ;
  IF CODE = ATTRIBUTE1 THEN
    BEGIN
      NEXTSYM ( WD , CODE ) ;
      IF CODE <> BLANK1 THEN GOTO 1 ;
      NEXTSYM ( WD , CODE ) ;
    END;
  IF CODE <> NAME1 THEN GOTO 1 ;
  (* ATTRIBUTE NAME WAS FOUND *)
  IF KIND = DOCUMENT1 THEN P1 := DOCATFIRST
```

```
                    ELSE P1 := COLATFIRST;
Q1 := P1 ;
IF P1 = NIL THEN
  BEGIN
   NEW( P1 ) ;
   P1@.NEXT := NIL ;
   P1@.NAME := WD ;
   IF KIND = DOCUMENT1 THEN DOCATFIRST := P1
                       ELSE COLATFIRST := P1;
  END
 ELSE
  BEGIN
   ENDREP := FALSE ;
   WHILE(NOT ENDREP) AND (P1 <> NIL) DO
    IF P1@.NAME = WD
     THEN ENDREP := TRUE
     ELSE BEGIN
           Q1 := P1 ;
           P1 := P1@.NEXT
          END;
   IF P1 = NIL THEN
    BEGIN
      NEW ( P1 ) ;
      Q1@.NEXT := P1 ;
      P1@.NEXT := NIL ;
      P1@.NAME := WD
    END
   END;
NEXTSYM ( WD , CODE ) ;
IF CODE = BLANK1 THEN NEXTSYM ( WD , CODE ) ;
IF CODE <> LEFTPAR1 THEN
  BEGIN
    1: DISP28('**** DEFINE - SYNTAX ERROR  ');
       DISPLAY ( NL )
  END
 ELSE
  BEGIN
   FIRST := NIL ;
   REPEAT
     NEXTSYM ( WD , CODE ) ;
     ENDREP := (CODE<>RIGHTPAR1)AND(CODE<>SEMICOLON1);
     IF ENDREP
      THEN IF(CODE<>BLANK1)AND(CODE<>NAME1)THEN GOTO 1
      ELSE IF CODE = NAME1
        THEN BEGIN
              NEW ( P ) ;
              P@.NAME := WD ;
              P@.BACK := P1 ;
              P@.NEXT := NIL ;
              IF FIRST = NIL THEN FIRST := P
                             ELSE Q@.NEXT := P ;
              Q := P ;
             END;
    UNTIL NOT ENDREP ;
   P1@.COM := FIRST ;
   END;
```

```
  IF NOT RETRIEVESW THEN WAS_A_SV := FALSE ;
END;


PROCEDURE ATTLIST ;
LABEL 1,2 ;
VAR
  W , W1 , W2 : ALFA28 ;
  CODE : IDENT ;
  P , P1 : ATTRIBPTR ;
  Q : COMPTR ;
  FOUND : BOOLEAN ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W , CODE ) ;
  IF CODE = DOCUMENT1 THEN P := DOCATFIRST
                      ELSE IF CODE = COLUMN1 THEN P := COLATFIRST
                                             ELSE GOTO 1 ;
  IF P = NIL THEN BEGIN
                  DISP8 ( 'NONE    ' ) ;
                  DISPLAY ( NL ) ;
                  GOTO 2 ;
                  END ;
  IF PRINTSW THEN
   BEGIN
    HEADER ; BLANKS(44) ;
    IF CODE = DOCUMENT1
     THEN WRITE8('DOCUMENT')
     ELSE WRITE8('COLUMN  ');
    WRITE8('ATTRIBUT'); WRITE2('ES');
    NEW_LINE ; NEW_LINE ;
   END ;
  NEXTSYM ( W1 , CODE ) ;
  IF CODE = BLANK1 THEN NEXTSYM ( W1 , CODE ) ;
  IF CODE = ATTRIBUTE1 THEN
                        BEGIN
                         NEXTSYM ( W , CODE ) ;
                         IF CODE = BLANK1 THEN NEXTSYM ( W1 , CODE ) ;
                        END;
  IF CODE = SEMICOLON1 THEN
    WHILE P <> NIL DO
     BEGIN
      IF PRINTSW THEN BEGIN WRITE28(P@.NAME);NEW_LINE END
                 ELSE DISP28NL (P@.NAME);
      Q:=P@.COM;
      WHILE Q<>NIL DO
       BEGIN IF PRINTSW THEN BEGIN
                              WRITE8(BLANK8);WRITE28(Q@.NAME);
                              NEW_LINE ;
                             END
                        ELSE BEGIN
                              DISP8(BLANK8);DISP28NL(Q@.NAME);
                             END;
             Q := Q@.NEXT;
      END;
```

```
      P := P@.NEXT ;
     END
    ELSE
     REPEAT
      P1 := P ;
      IF CODE <> NAME1 THEN GOTO 1 ;
      FOUND := FALSE ;
      WHILE (P<>NIL) AND (NOT FOUND) DO
       IF P@.NAME = W1 THEN
         BEGIN
          FOUND := TRUE ;
          IF PRINTSW THEN BEGIN WRITE28(W1);NEW_LINE END
                     ELSE DISP28NL(W1);
          Q := P@.COM ;
          WHILE Q <> NIL DO
           BEGIN
            IF PRINTSW THEN BEGIN WRITE8(BLANK8);WRITE28(Q@.NAME);
                                  NEW_LINE ;
                            END
                       ELSE BEGIN DISP8(BLANK8);DISP28NL(Q@.NAME);
                            END;
            Q := Q@.NEXT ;
           END;
         END
        ELSE P := P@.NEXT ;
       IF NOT FOUND THEN
          BEGIN
           DISP28BL(W1); DISP8(' IS NOT ');DISP8('DEFINED ');
           DISPLAY(NL);
          END;
       P := P1 ;
       NEXTSYM ( W , CODE ) ;
       IF CODE = BLANK1 THEN NEXTSYM ( W , CODE ) ;
     UNTIL CODE = SEMICOLON1 ;
     GOTO 2 ;
     1 : DISP28 ('**** ATLIST - SYNTAX ERROR  ');
         DISPLAY ( NL ) ;
     2 :
END;


(*************************************************************************)
(*                                                                       *)
(*                R    E    M    O    V    E                            *)
(*                                                                       *)
(*************************************************************************)


PROCEDURE REMOVE ;
LABEL 1 ;
VAR
  W : ALFA28 ;
  CODE, CODE1 : IDENT ;
  FIRST, T, T1 : ATTRIBPTR ;
  FOUND : BOOLEAN ;
  R, R1 : ATRPTR ;
```

```
  P : DOCPTR ;
  CH : CHAR ;
  Q : COLPTR ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W , CODE1 ) ;
  IF(CODE1<>DOCUMENT1)AND(CODE1<>COLUMN1) THEN GOTO 1 ;
  NEXTSYM ( W , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W , CODE ) ;
  IF CODE = ATTRIBUTE1 THEN
   BEGIN
    NEXTSYM ( W , CODE ) ;
    IF CODE <> BLANK1 THEN GOTO 1 ;
    NEXTSYM ( W , CODE ) ;
   END ;
  IF CODE <> NAME1 THEN
   BEGIN 1 : DISP28NL('**** REMOVE - SYNTAX ERROR  ');
   END
  ELSE
   BEGIN
    IF CODE1 = DOCUMENT1
     THEN FIRST := DOCATFIRST
     ELSE FIRST := COLATFIRST ;
    T := FIRST ;
    T1 := T ;
    FOUND := FALSE ;
    WHILE (NOT FOUND) AND (T<>NIL) DO
     IF T@.NAME = W
      THEN FOUND := TRUE
      ELSE
       BEGIN
        T1 := T ;
        T := T@.NEXT ;
       END ;
    IF NOT FOUND
     THEN DISP28NL('ERROR - ATTRIBUTE NOT FOUND ')
     ELSE
      BEGIN
        (* REMOVE OCCURANCES OF THE ATTRIBUTES *)
       FOR CH := 'A' TO 'Z' DO
        BEGIN
         P := DOCTABLE [ CH ] ;
         WHILE P <> NIL DO
         BEGIN
         IF CODE1 = DOCUMENT1
          THEN
           BEGIN
            R := P@.ATRP ;
            R1 := R ; FOUND := FALSE ;
            WHILE (R<>NIL)AND(NOT FOUND) DO
             IF R@.COMPT@.BACK = T
              THEN FOUND := TRUE
              ELSE BEGIN
                   R1 := R ;
```

```
                    R := R@.NEXT ;
                    END;
                IF FOUND THEN
                IF R1 = P@.ATRP
                 THEN P@.ATRP := R@.NEXT
                 ELSE R1@.NEXT := R@.NEXT ;
               END
              ELSE
               BEGIN
                Q := P@.COLU ;
                WHILE Q<>NIL DO
                 BEGIN
                  R := Q@.ATRP ; R1:=R; FOUND := FALSE ;
                  WHILE (R<>NIL)AND(NOT FOUND) DO
                   IF R@.COMPT@.BACK = T
                    THEN FOUND := TRUE
                    ELSE BEGIN
                          R1 := R ;
                          R := R@.NEXT ;
                         END;
                  IF FOUND THEN
                  IF R1 = Q@.ATRP
                   THEN Q@.ATRP := R@.NEXT
                   ELSE R1@.NEXT := R@.NEXT ;
                  Q := Q@.NEXT
                 END ;
               END ;
            P := P@.NEXT
           END
          END;
         IF T1 = FIRST
          THEN
           IF CODE1 = DOCUMENT1 THEN DOCATFIRST := T@.NEXT
                                ELSE COLATFIRST := T@.NEXT
          ELSE
           T1@.NEXT := T@.NEXT ;
      END;
    END;
END;


(*****************************************************************************)
(*                                                                           *)
(*        S E L E C T                    R O U T I N E S                     *)
(*                                                                           *)
(*****************************************************************************)


PROCEDURE SELECTSRCH(VAR S:STATUS;VAR R:COMPTR;VAR FOUND:BOOLEAN);
VAR
  W1, W2 : ALFA28 ;
  CODE : IDENT ;
BEGIN
  S := WASNOTOK ;
  FOUND := FALSE ;
  NEXTSYM ( W1 , CODE ) ;
```

```
  IF CODE = BLANK1 THEN
   BEGIN
    NEXTSYM ( W1 , CODE ) ;
    IF CODE = NAME1 THEN
     BEGIN
      NEXTSYM (W2 , CODE ) ;
      IF CODE = BLANK1 THEN NEXTSYM ( W2 , CODE ) ;
      IF CODE = PERIOD1 THEN
       BEGIN
        NEXTSYM ( W2 , CODE ) ;
        IF CODE = BLANK1 THEN NEXTSYM ( W2 , CODE ) ;
        IF CODE = NAME1 THEN
         BEGIN
          SEARCHATR ( DOCATFIRST , W1 , W2 , FOUND , R );
          S := WASOK ;
         END;
      END;
     END
   END
END;


PROCEDURE SELECTCOL ( W : ALFA28 ; T1 : COLOCPTR ; VAR NLINES : SHORTINT ) ;
(* FIND AND PRINT IF COLUMN W SATISFIES THE ATTRIBUTES SPECIFIED BY WITH  *)
(* AND WITHOUT ARRAYS. T1 POINTS TO BEGINNING OF THE COLUMN OCCURANCE LIST*)
VAR
  I : SHORTINT ;
  A, B : ARRAY [ 1 .. ARRAY_SIZE ] OF SHORTINT ;
  R : ATRPTR ;
  T : COLOCPTR ;
  P : DOCPTR ;
  FOUND : BOOLEAN ;
BEGIN
  FOR I := 1 TO ARRAY_SIZE DO
   BEGIN
    A [ I ] := 0 ;
    B [ I ] := 0 ;
   END ;
  T := T1 ;
  WHILE T <> NIL DO
   BEGIN   (* SCAN ALL OCCURANCES OF COLUMN W *)
    P := T@.PDOC ;
    R := P@.ATRP ;
    WHILE R <> NIL DO
     BEGIN
      IF INDWI > 0 THEN
       FOR I := 1 TO INDWI DO
        IF R@.COMPT = WITHAR [ I ] THEN A [ I ] := 1 ;
      IF INDWO > 0 THEN
       FOR I := 1 TO INDWO DO
        IF R@.COMPT = WITHOUTAR [ I ] THEN B [ I ] := 1 ;
      R := R@.NEXT ;
     END ;
    T := T@.NEXT ;
   END ;
  FOUND := TRUE ;
```

```
  IF INDWI > 0 THEN
   FOR I := 1 TO INDWI DO
    IF A [ I ] = 0 THEN FOUND := FALSE ;
  IF FOUND AND ( INDWO > 0 ) THEN
   FOR I := 1 TO INDWO DO
    IF B [ I ] = 1 THEN FOUND := FALSE ;
  IF FOUND THEN
   IF PRINTSW
    THEN
     BEGIN
      WRITE28 ( W ) ;
      NEW_LINE ;
     END
    ELSE
    BEGIN
     NLINES := NLINES + 1 ;
     IF NLINES MOD SCREEN_LENGTH = 0 THEN NEXT_SCREEN ;
     DISP28NL(W) ;
    END;
END;


PROCEDURE SELECT ;
LABEL 1 ;
CONST
  MESSAGE1 = 'ATTRIB/COMPON NOT FOUND      ' ;
  MESSAGE2 = 'WITH / WITHOUT  OVERFLOW     ' ;
VAR
  W : ALFA28 ;
  S : STATUS ;
  R : COMPTR ;
  ERROR, FOUND : BOOLEAN ;
  CODE : IDENT ;
  Q : COLTBPTR ;
  CH : CHAR ;
  T : COLOCPTR ;
  NLINES, I : SHORTINT ;
BEGIN
  NEXTSYM ( W , CODE ) ;
  IF CODE <> BLANK1 THEN GOTO 1 ;
  NEXTSYM ( W , CODE ) ;
  IF CODE = COLUMN1 THEN
   BEGIN
    NEXTSYM ( W , CODE ) ;
    IF CODE <> BLANK1 THEN GOTO 1 ;
    NEXTSYM ( W , CODE ) ;
   END ;
  INDWI := 0 ;
  INDWO := 0 ;
  ERROR := FALSE ;
  REPEAT
   CASE CODE OF
    BLANK1    : ;
    SEMICOLON1: ;
    WITH1     : BEGIN
                  INDWI := INDWI + 1 ;
```

```
                IF INDWI > ARRAY_SIZE
                 THEN
                  BEGIN
                   DISP28NL ( MESSAGE2 ) ; ERROR := TRUE
                  END
                 ELSE
                  BEGIN
                   SELECTSRCH ( S , R , FOUND ) ;
                   IF S = WASOK
                     THEN
                     IF FOUND THEN WITHAR [ INDWI ] := R
                              ELSE BEGIN
                               DISP28NL( MESSAGE1 ) ;
                               ERROR := TRUE ;
                                   END
                     ELSE GOTO 1 ;
                   END ;
                END;
   WITHOUT1   : BEGIN
                 INDWO := INDWO + 1 ;
                 IF INDWO > ARRAY_SIZE
                  THEN
                   BEGIN
                    DISP28NL ( MESSAGE2 ) ;
                    ERROR := TRUE ;
                   END
                  ELSE
                   BEGIN
                    SELECTSRCH ( S , R , FOUND ) ;
                    IF S = WASOK
                     THEN
                      IF FOUND
                       THEN
                        WITHOUTAR [ INDWO ] := R
                       ELSE
                        BEGIN
                         DISP28NL ( MESSAGE1 ) ; ERROR := TRUE ;
                        END
                      ELSE
                       GOTO 1 ;
                    END;
                 END;
   ELSE       : BEGIN 1 :
                 DISP28NL('**** SELECT - SYNTAX ERROR  ');
                 ERROR := TRUE
                END;
  END;
  IF NOT ERROR THEN NEXTSYM ( W , CODE ) ;
 UNTIL ERROR OR ( CODE = SEMICOLON1 ) ;
IF NOT ERROR THEN
BEGIN
 IF PRINTSW THEN
  BEGIN
   HEADER ; BLANKS ( 44 ) ;
   WRITE28(' SELECT   LISTING           ') ;
   NEW_LINE ; NEW_LINE ;
```

```
      IF INDWO > 0 THEN
       FOR I := 1 TO INDWO DO
        BEGIN
         BLANKS ( 40 ) ; WRITE8('WITHOUT ');
         WRITE28BL(WITHOUTAR[I]@.BACK@.NAME);
         WRITE2('. ');
         WRITE28BL(WITHOUTAR[I]@.NAME); NEW_LINE ;
        END;
      IF INDWI > 0 THEN
       FOR I := 1 TO INDWI DO
        BEGIN
         BLANKS ( 40 ) ; WRITE8('WITH    ');
         WRITE28BL(WITHAR[I]@.BACK@.NAME);
         WRITE2('. ');
         WRITE28BL(WITHAR[I]@.NAME); NEW_LINE ;
        END;
      NEW_LINE ;
    END ;
   NLINES := 0 ;
   FOR CH := 'A' TO 'Z' DO
    BEGIN
     Q := COLTABLE [ CH ] ;
     WHILE Q <> NIL DO
      BEGIN
       T := Q@.COLLIST ;
       SELECTCOL ( Q@.NAME , T , NLINES ) ;
       Q := Q@.NEXT ;
      END;
    END;
  END;
END;

(*****************************************************************************)
(*                                                                           *)
(*          G E N E R A L                    R O U T I N E S                 *)
(*                                                                           *)
(*****************************************************************************)


PROCEDURE NEXT_SCREEN ;
VAR
   CH : CHAR ;
BEGIN
   ACCEPT ( CH ) ;
   WHILE CH <> NL DO
    ACCEPT ( CH ) ;
END;


PROCEDURE GETCOMD  ;
VAR CH : CHAR ;
    I : SHORTINT ;
BEGIN
   S := WASOK ;
   CURRENT := 1 ;
```

```
  LASTCHAR := 0 ;
  REPEAT
    ACCEPT ( CH ) ;
    CASE CH OF
      '&' : LASTCHAR := 0 ;
      '@' : BEGIN
              LASTCHAR := LASTCHAR - 1 ;
              IF LASTCHAR < 0 THEN LASTCHAR := 0
            END;
      ',' : BEGIN
                LASTCHAR := LASTCHAR + 1 ;
                COMMAND [ LASTCHAR ] := ' '
            END;
      NL  : BEGIN
              LASTCHAR := LASTCHAR + 1 ;
              COMMAND [ LASTCHAR ] := ' '
            END;
      ELSE: BEGIN
              LASTCHAR := LASTCHAR + 1 ;
              COMMAND [ LASTCHAR ] := CH
            END;
    END;
  UNTIL ( CH = ';' ) OR ( LASTCHAR >= NCHAR );
  IF CH <> ';' THEN
    BEGIN
      S := WASNOTOK ;
      DISP28('ERROR - COMMAND IS TOO LONG ');
      DISPLAY(NL)
    END
    ELSE IF LASTCHAR < NCHAR THEN
         FOR I := LASTCHAR + 1 TO NCHAR DO
             COMMAND [ I ] := ';' ;
    WHILE CH <> NL DO ACCEPT ( CH ) ;    " GET RID OF REMAINING NL "
END;


PROCEDURE NEXTWORD ( VAR W : ALFA28 ) ;
VAR
  NCH : SHORTINT ;
  CH : CHAR ;
  ISLETTER : BOOLEAN ;
BEGIN
  W := BLANK28 ;
  NCH := 0 ;
  REPEAT
    CH := COMMAND [ CURRENT ] ;
    ISLETTER := CH IN LETTERS ;
    IF ISLETTER THEN
      BEGIN
        NCH := NCH + 1 ;
        W [ NCH ] := CH ;
        CURRENT := CURRENT + 1
      END;
  UNTIL (NOT ISLETTER) OR (NCH = NAMESIZE) OR (CURRENT > LASTCHAR);
END;
```

```
PROCEDURE NEXTSYM ;
CONST
  ATTRIBUTE2 = 'ATTRIBUTE                    ';
  ATTRIBUTE3 = 'AT                           ';
  PREPARE2   = 'PREPARE                      ';
  PREPARE3   = 'PE                           ';
  TEACH2     = 'TEACH                        ';
  TEACH3     = 'TE                           ';
  SAVE2      = 'SAVE                         ';
  SAVE3      = 'SV                           ';
  LIST2      = 'LIST                         ';
  LIST3      = 'LS                           ';
  XREF2      = 'XREF                         ';
  XREF3      = 'XR                           ';
  DOCUMENT2  = 'DOCUMENT                     ';
  DOCUMENT3  = 'DC                           ';
  COLUMN2    = 'COLUMN                       ';
  COLUMN3    = 'CL                           ';
  FIND2      = 'FIND                         ';
  FIND3      = 'FI                           ';
  KEY2       = 'KEY                          ';
  KEY3       = 'KY                           ';
  SIMILAR2   = 'SIMILAR                      ';
  SIMILAR3   = 'SM                           ';
  TO2        = 'TO                           ';
  FROM2      = 'FROM                         ';
  END2       = 'END                          ';
  END3       = 'EN                           ';
  ADD2       = 'ADD                          ';
  ADD3       = 'AD                           ';
  RENAME2    = 'RENAME                       ';
  RENAME3    = 'RN                           ';
  AS2        = 'AS                           ';
  IN2        = 'IN                           ';
  DELETE2    = 'DELETE                       ';
  DELETE3    = 'DL                           ';
  DEFINE2    = 'DEFINE                       ';
  DEFINE3    = 'DF                           ';
  PRINT2     = 'PRINT                        ';
  PRINT3     = 'PR                           ';
  YES2       = 'YES                          ';
  YES3       = 'Y                            ';
  NO2        = 'NO                           ';
  NO3        = 'N                            ';
  ATLIST2    = 'ATLIST                       ';
  ATLIST3    = 'AL                           ';
  RETRIEVE2  = 'RETRIEVE                     ';
  RETRIEVE3  = 'RT                           ';
  RUNNAME2   = 'RUNNAME                      ';
  RUNNAME3   = 'RU                           ';
  SELECT2    = 'SELECT                       ';
  SELECT3    = 'SL                           ';
  WITH2      = 'WITH                         ';
  WITHOUT2   = 'WITHOUT                      ';
  REMOVE2    = 'REMOVE                       ';
```

```
  REMOVE3    = 'RM                              ';
VAR
  CH : CHAR ;
BEGIN
  CODE := OTHER1 ;
  CH := COMMAND [ CURRENT ] ;
  IF CH IN LETTERS THEN
  BEGIN
    NEXTWORD ( WD ) ;
    CODE := NAME1 ;
    CASE CH OF
     'A' : IF(WD=ADD2)OR(WD=ADD3) THEN CODE:= ADD1
           ELSE IF (WD=ATTRIBUTE2)OR(WD=ATTRIBUTE3)THEN CODE:= ATTRIBUTE1
           ELSE IF(WD=ATLIST2)OR(WD=ATLIST3)THEN CODE:=ATLIST1
           ELSE IF WD=AS2 THEN CODE:=AS1;
     'B' : ;
     'C' : IF(WD=COLUMN2)OR(WD=COLUMN3)THEN CODE:=COLUMN1;
     'D' : IF(WD=DELETE2)OR(WD=DELETE3)THEN CODE:=DELETE1
           ELSE IF(WD=DEFINE2)OR(WD=DEFINE3)THEN CODE:=DEFINE1
           ELSE IF(WD=DOCUMENT2)OR(WD=DOCUMENT3)THEN CODE:=DOCUMENT1;
     'E' : IF(WD=END2)OR(WD=END3)THEN CODE:=END1;
     'F' : IF(WD=FIND2)OR(WD=FIND3)THEN CODE:=FIND1
           ELSE IF WD = FROM2 THEN CODE := TO1 ;
     'G' : ;
     'H' : ;
     'I' : IF WD=IN2 THEN CODE:=IN1;
     'J' : ;
     'K' : IF(WD=KEY2)OR(WD=KEY3)THEN CODE:=KEY1;
     'L' : IF(WD=LIST2)OR(WD=LIST3)THEN CODE:=LIST1;
     'M' : ;
     'N' : IF(WD=NO2)OR(WD=NO3)THEN CODE:=NO1;
     'O' : ;
     'P' : IF(WD=PREPARE2)OR(WD=PREPARE3)THEN CODE:=PREPARE1
           ELSE IF(WD=PRINT2)OR(WD=PRINT3)THEN CODE:=PRINT1;
     'Q' : ;
     'R' : IF(WD=RENAME2)OR(WD=RENAME3)THEN CODE:=RENAME1
           ELSE IF(WD=RUNNAME2)OR(WD=RUNNAME3)THEN CODE:=RUNNAME1
           ELSE IF(WD=REMOVE2)OR(WD=REMOVE3)THEN CODE:=REMOVE1
           ELSE IF(WD=RETRIEVE2)OR(WD=RETRIEVE3)THEN CODE:=RETRIEVE1;
     'S' : IF(WD=SIMILAR2)OR(WD=SIMILAR3)THEN CODE:=SIMILAR1
           ELSE IF (WD=SELECT2)OR(WD=SELECT3)THEN CODE:=SELECT1
           ELSE IF(WD=SAVE2)OR(WD=SAVE3)THEN CODE:=SAVE1;
     'T' : IF(WD=TEACH2)OR(WD=TEACH3)THEN CODE:=TEACH1
           ELSE IF WD=TO2 THEN CODE:=TO1;
     'U' : ;
     'V' : ;
     'W' : IF WD = WITH2 THEN CODE := WITH1
           ELSE IF WD = WITHOUT2 THEN CODE := WITHOUT1 ;
     'X' : IF(WD=XREF2)OR(WD=XREF3)THEN CODE:=XREF1;
     'Y' : IF(WD=YES2)OR(WD=YES3)THEN CODE:=YES1;
     'Z' : ;
    ELSE : IF WD[1] IN [ '0' .. '9' ] THEN CODE := OTHER1 ;
    END
 END
   ELSE
     CASE CH OF
```

```
      ' ' : BEGIN
              REPEAT
               CURRENT := CURRENT + 1
              UNTIL COMMAND [ CURRENT ] <> ' ';
              CODE := BLANK1
            END;
      '.' : BEGIN
              CODE := PERIOD1 ;
              CURRENT := CURRENT + 1
            END;
      ';' : CODE := SEMICOLON1 ;
      '(' : BEGIN
              CODE := LEFTPAR1 ;
              CURRENT := CURRENT + 1
            END;
      ')' : BEGIN
              CODE := RIGHTPAR1 ;
              CURRENT := CURRENT + 1
            END;
       ELSE: ;
     END; (* OF CASE STATEMENT *)
END;


PROCEDURE MAINLOOP ;
LABEL 1;
VAR WD : ALFA28;
    CODE : IDENT ;
    S : STATUS ;
BEGIN
  REPEAT
   DISP8('COMMAND:'); DISPLAY(NL);
   PRINTSW := FALSE ;
   GETCOMD( S ) ;
   NEXTSYM( WD , CODE ) ;
   1 : CASE CODE OF
    BLANK1 : BEGIN
             NEXTSYM(WD , CODE);
             GOTO 1;
             END;
      SEMICOLON1 : ;
      TEACH1 : TEACH ;
      SAVE1  : SAVE ;
      LIST1  : LIST ;
      XREF1  : XREF ;
      FIND1  : FIND ;
      KEY1   : KEY ;
      PREPARE1 : PREPARE ;
      END1   : IF NOT WAS_A_SV THEN
                BEGIN
                 REPEAT
                  DISP28('DATA WAS NOT SAVED,TYPE YES/');
                  DISP28NL('NO TO CONTINUE WITH END     ');
                   GETCOMD ( S ) ; NEXTSYM ( WD , CODE ) ;
                  UNTIL ( CODE = YES1 ) OR ( CODE = NO1 ) ;
                  IF CODE = YES1 THEN CODE := END1 ;
```

```
                END ;
      ADD1    : ADD ;
      RENAME1 : RENAME ;
      RUNNAME1: RUNNAME ;
      DELETE1 : DELETE ;
      DEFINE1 : DEFINE ;
      PRINT1  : BEGIN
                  PRINTSW := TRUE ; NEXTSYM ( WD , CODE ) ;
                  GOTO 1 ;
                END;
     DOCUMENT1 : DOCUMENTS ;
       COLUMN1 : COLUMNS ;
      ATLIST1  : ATTLIST ;
      SELECT1  : SELECT ;
     RETRIEVE1 : RETRIEVE ;
     REMOVE1 : REMOVE ;
      ELSE    : BEGIN
                  DISP28('ERROR - UNDEFINED COMMAND   ');
                  DISP28(WD); DISPLAY(NL)
                END;
      END; (* OF CASE STATEMENT *)
    UNTIL CODE = END1;
END;

(*****************************************************************************)
(*                                                                           *)
(*           M  A  I  N                P  R  O  G  R  A  M                  *)
(*                                                                           *)
(*****************************************************************************)


BEGIN
    LETTERS := [ 'A' .. 'Z' , '0' .. '9' , '_'  ] ;
    DIGITS := [ '0' .. '9' ] ;
    COLATFIRST := NIL ;
    DOCATFIRST := NIL ;
    WAS_A_SV := TRUE ;
    SVC2FDAT ( DATE ) ;
    RETRIEVESW := FALSE ;
    NAME_OF_RUN := BLANK28 ;
    LINES := 0 ;
    PAGES := 0 ;
    WAS_A_RT := FALSE ;
    FOR CH := 'A' TO 'Z' DO
       BEGIN
         DOCTABLE [ CH ] := NIL ;
         COLTABLE [ CH ] := NIL
       END;
    DISP28('TYPE TEACH FOR LIST OF COMMA');
    DISP8('NDS     '); DISPLAY ( NL ) ;
    MAINLOOP ;
END.
```

BIBLIOGRAPHY

Bernstein,P.A. Synthesizing third normal form relations from functional dependencies, ACM Transactions on Database Systems 1,4 (Dec 1976), 277-298.

Fisher,P.S. - A new approach to data base design, Unpublished paper, Kansas State University, 1979.

Hollist,P.J. - An integrated data base design methodology, Masters report, Kansas State University, 1980.

Jensen,K. and Wirth,N. - Pascal user's manual and report, Springer - Verlag 1973.

A SYSTEM FOR AUTOMATIC GENERATION OF RELATIONAL DATA BASES

BY

MEIR COHEN

B.S., TEL AVIV UNIVERSITY, ISRAEL, 1979

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1981

A B S T R A C T

The usefulness of a data base management system is directly related to the way in which the actual data base was designed. The design process may be divided into two phases, the logical design and the physical design. This report deals with a logical design method that uses the information found on any organization's forms as a basis to the complete design process. An implementation of an interactive system that automates the logical design process is described. Finally, an extensive example is provided, demonstrating the design process and the use of the interactive system.