A SOFTWARE SYSTEM FOR THE DATA RETRIEVAL SUBLANGUAGE FOR A DYNAMIC DBMS

A MASTER THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE COMMITTEE ON THE FACULTY OF ENGINEERING
OF MIDDLE EAST TECHNICAL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF
MASTER OF SCIENCE

by

Alev Elçi

January 1982

I certify that I have read this thesis and in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

.....

Supervisor

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

.....

Chairman of the Department

Examining Committee in Charge

Asst.Prof.Dr Abdullah Uz Tansel .....

Asst.Prof.Dr Müslim Bozyiğit .....

Asst.Prof.Dr Neşe Yalabık .....

Asst.Prof.Dr Halil Başoğlu .....

Asst.Prof. Bülent Epir .....

ABSTRACT

A SOFTWARE SYSTEM
FOR THE DATA RETRIEVAL SUBLANGUAGE
FOR A DYNAMIC DBMS

ELÇİ, Alev.
M.Sc. in Computer Eng.
Supervisor: Dr Halil Başoğlu
January 1982; viii+ 118 pages

The Dynamic Database Management System,DDS, designed in METU, is a database management system enabling the user to dynamically define and utilize the database of his/her own needs. DDS has a nonprocedural, relational, and high-level data language whose syntax is based on the natural structure of Turkish. Its data retrieval sublanguage has been proved to be relationally complete.

This research work is a part of the software development effort to implement DDS. The purpose of this work is to design and implement a software package for the data retrieval sublanguage of DDS. The sublanguage is used in finding answers in the database to the queries of various users. It consists of four kinds of retrieval statements, namely, unconditional, conditional, set-conditional, and keyed. There are also auxiliary statements for additional operations on the retrieved data items.

KEYWORDS: Dynamic database management system, data retrieval sublanguage, relational approach, Turkish-based data language.

ÖZET

DİNAMİK BİR VERİTABANI YÖNETİM SİSTEMİNİN VERİ ÇEKME ALTDİLİ İÇİN BİR YAZILIM SİSTEMİ

ELÇİ, Alev.
Bilgisayar Mühendisliği
Yüksek Lisans Tezi
Yönetici:Dr Halil Başoğlu
Ocak 1982; viii+ 118 sayfa

Dinamik Veritabanı Yönetim Sistemi, DVS, kullanıcının gereksinmelerinin veritabanını dinamik olarak tanımlanması ve kullanmasına yarayan ODTÜ'de tasarlanmış bir veritabanı yönetim sistemidir. DVS veri dili, yöntemsel olmayan, ilişkisel, ve yüksek düzeyli bir dildir; ve sözdizimi Türkçe'nin doğal yapısına uyumludur. Veri çekme altdilinin ilişkisel bütünlüğü kanıtlanmıştır.

Bu araştırma çalışması, DVS'yi uygulamak için yapılan yazılım geliştirme çalışmasının bir parçasıdır. Bu çalışmanın amacı, DVS veri çekme altdili için bir yazılım paketini tasarımlamak ve gerçekleştirmektir. Altdil, değişik kullanıcıların sorgularına veritabanında yanıtlar bulmakta kullanılır; koşulsuz, koşullu, küme koşullu, ve anahtarlı gibi dört tür çekme deyiminden oluşur. Çekilen veriler üzerinde yapılmak istenecek ek işlemler için de yardımcı deyimleri vardır.

ANAHTAR SÖZCÜKLER: Dinamik veritabanı yönetim sistemi, veri çekme altdili, ilişkisel yaklaşım, Türkçe tabanlı veri dili.

ACKNOWLEDGEMENTS

I would like to thank Dr Y.Uçkan for his help in starting this work and to Dr H.Başoğlu for his support and patience in completing it. I am grateful to my husband Atilla for his support and patience at home. Also I thank the personnel of Koç-Burroughs, Ankara Division, especially Mr. C.Nayır, ÖSYM and MTA Computer Center personnel for the understanding they have shown.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1. General

A database consists of a collection of data which is stored on a secondary storage media, and is being used by application programs in retrieval, updating, insertion, and deletion operations (1, 2).

The most important reason for using databases is the centralized control of data. The advantages of having centralized control of data can be summarized as :

a. There will be a reduction of redundancy in the data stored; in other words the data will be stored once and will be shared by several applications. The removal of redundancy results in a net saving of the storage space; in addition, it eliminates the inconsistency of the update operations.

b. There will be a standard in data representation so that data interchange between different installations will be possible.

c. A security system will be used to restrict the access to the database by unauthorized personnel.

d. Data integrity is maintained by using certain procedures before storage operations so that incorrect data will not be stored.

e. Data independence will be obtained. This requirement arises from different applications needing different views of the same data, storage structure and access strategy.

Because of the need for integration of large files into a database, and the need for many users and application programs to access it efficiently, the database management system (DBMS) concept was introduced. In this respect generalised file management systems were limited by single-file structures of their host operating systems, and conventional file structures were inadequate for complex queries in database systems (1).

A DBMS must provide facilities for its user, the database administrator (DBA) and the others who will access a database according to a view (4). This view is patterned according to one of generalised data models (see section 1.2). DBA is responsible of creating and maintaining the physical storage structures, whereas, the users are interested in the logical aspects of data organization and access.

A user accesses the data and its relationships available in a 'view' via a data language. The users may vary in data language capabilities and view requirements.

DBMS must meet the following requirements in order to give service to the users :

a) Users must be allowed to communicate with the DBMS with a language close to their natural language.

b) DBMS must supply the users with their information needs.

There are two types of languages that are supported by a DBMS. These are data definition language (DDL) and data manipulation language (DML). DDL is provided for DBA, and is used to describe schemas and subschemas (4).

Data manipulation facility, which is in the scope of this study, is the main interface between a user and the DBMS.

The operations permitted on the database can be invoked by using the DML commands that retrieve and transfer data to the application program, and add, change, and delete data. The commands may also have parameter lists, including the name of variables that will be used in the operation. There are three types of DML commands : control, retrieval, and modification. Control commands identify an application, determine the database it wants to access, allocate buffers and files, and determine the commands the application may use. Retrieval commands select data from the database and perform the required operations on it. Modification commands do the insertion, update, and deletion operations on the database.

Characteristics of a DML can be summarized as follows:

a) DML must be natural and easy to use, so that the user can easily understand the semantics of the operations permitted on the database.

b) It should be precise and complete, in other words, when a data or a relationship on the data is specified there must be no ambiguity.

c) DML should facilitate an efficient implementation.

In the following section an overview of the data languages used in different database management systems will be given.

1.2. An Overview Of Data Languages In Database Management Systems

The architecture for database systems is divided into three general levels (2, 4) internal, conceptual, and external. Internal level is the way in which data is actually stored (physical storage); and external level is the way in which the data is seen by the users; and conceptual is the level between external and internal

levels. The conceptual model (data model) is a pattern according to which data is logically organized. The logical units of data and the relationship among them build up the data model.

It is seen from the above definitions that the data model is the most important part of a database system. There are three main approaches handling the relationships among data in different ways : the relational, the hierarchical, and the network.

The relational data model is a formal model for representing relationships among attributes of an entity set and the associations between entity sets (4). The relations are formed as tables of data, where the rows are called tuples, and the columns are called attributes. These relations may be considered as being mathematical. Their difference from the mathematical relations is that the former can be changed from time to time; some tuples may be deleted, inserted, or updated. The relational data language has the facility of allowing the user to construct new relations from the existing ones using the relational operators. Some known relational data languages are ALPHA which is based on relational calculus (2, 4), SEQUEL- Structural English Query Language (2, 4), and Query By Example (4).

In the hierarchical data model, the data is represented by simple tree structures. Every node in the hierarchical tree represents a record type, and two record types are connected by links. The disjoint trees are collected to form a hierarchical database with a unique root. There can be a varying number of occurrences of each record type at each level in a hierarchical database tree. Because of the links between record types in this model, insertion and deletion of records need special consideration. When a new record is to be inserted in to the database, it must be connected to a parent record also (except the root record). IMS

(Information Management System) is one of the most known database systems based on hierarchical approach which provides a record-at-a-time data language DL/I (2, 4).

The third approach is the network data model. The data is represented by record types and links as in hierarchical model. The difference is that, a network can have an unlimited number of superiors not one as in hierarchical. The important examples of network systems are Burroughs' DMS II (14, 15, 16, 17), and DBTG which is provided by the CODASYL Data Base Task Group (2).

For an application, the database is organised according to one of the data models defined above, and it is accessed using a data language. A data language is a set of operations which permit the user to access the data, that is organised by a data model (4). There are two types of data language operations: retrieval, and modification. Retrieval operations are for getting data values from the database, and modification operations are for changing the value of data.

There are various types of data languages in a DBMS (2, 1). A host data language is embedded in a procedural language such as COBOL, PL/I, or FORTRAN which are called host languages. The data manipulation language (DML) or data sublanguage (DSL) consists of statements written in a host language. When one of the statements is used, the corresponding handler routine of the DBMS gets invoked; and the resultant data is returned to the host language. DML or DSL may be an extension of the host language or a separate language of the DBMS.

A query language can be used for both handling and manipulating data. The programming language is a nonprocedural, English-like, easy to learn language so that nonprogrammers may easily use it.

The data language of the Dynamic Database Management System (DDS) is implemented as a stand alone query language, although Uçkan (7) specified that it may also be used embedded in a host language.

The data languages may be further qualified as general and application oriented (4). The general data languages answer different questions of different users. Application oriented languages can only be used for specific applications and queries.

Some general definitions about data model approaches and data languages were given above. In the following sections, detailed information on the representative data languages of each data model will be given. All the examples refer to the UNIVERSITY database (7) whose description in the three models are given in figures 1.1 through 1.3.

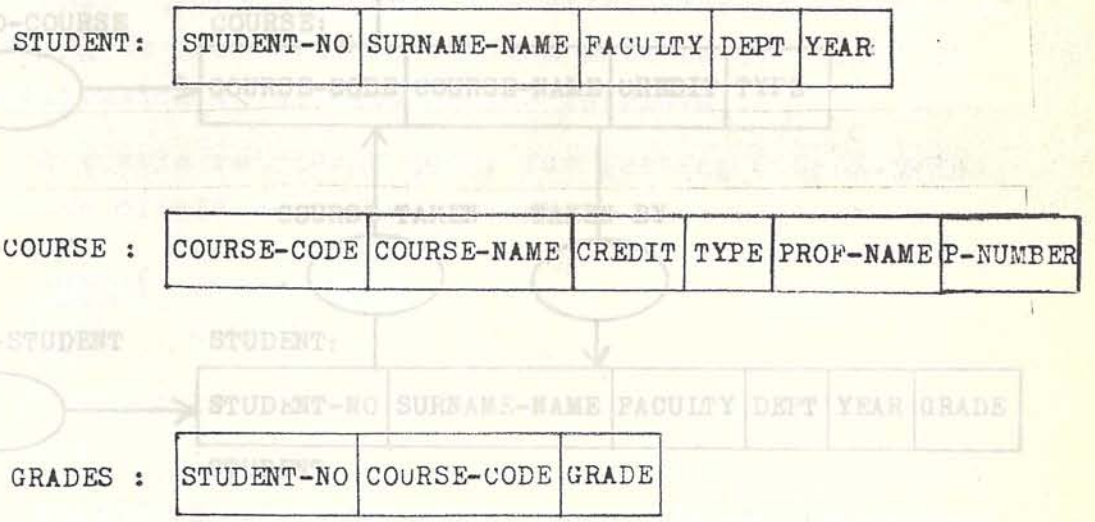

Figure 1.1 Definition of the database UNIVERSITY in relational model.

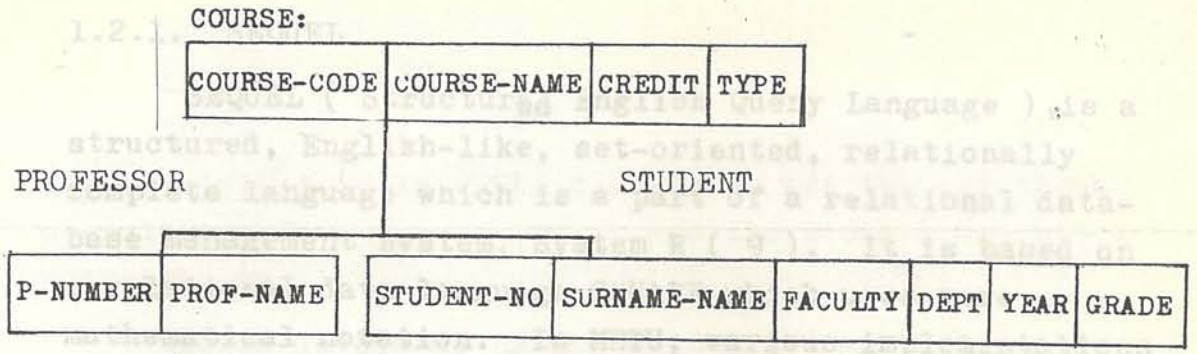

Figure 1.2 Definition of the database UNIVERSITY in the hierarchical model.

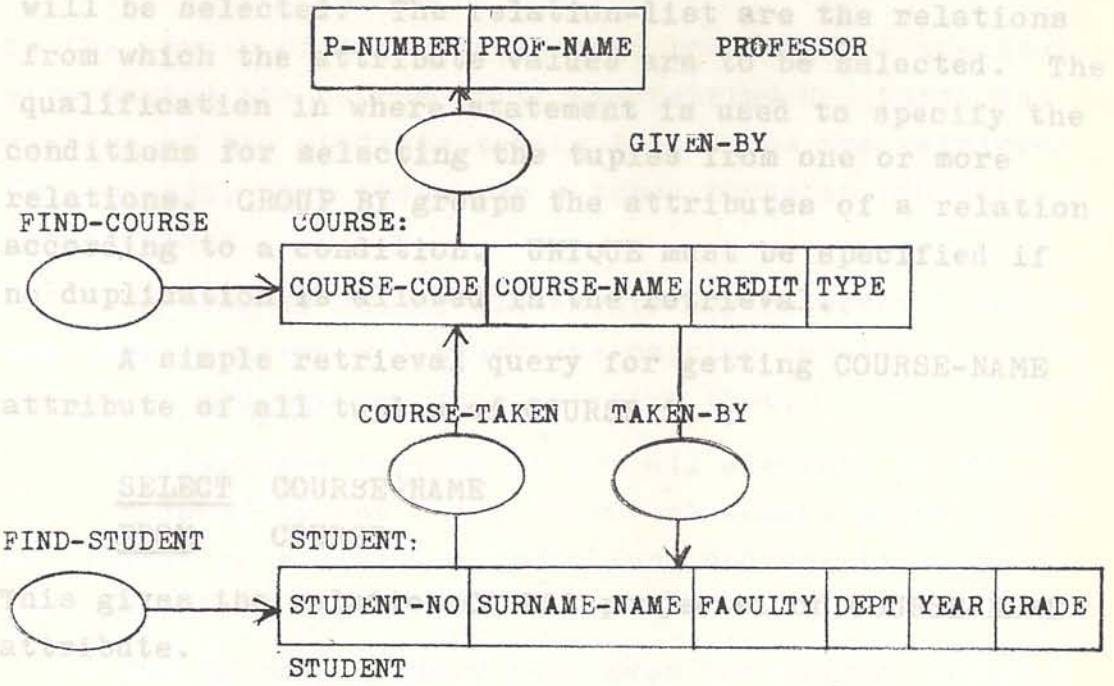

Figure 1.3 Definition of the database UNIVERSITY in the network model.

1.2.1. SEQUEL

SEQUEL (Structured English Query Language) is a structured, English-like, set-oriented, relationally complete language which is a part of a relational database management system, System R (9). It is based on a relational data language SQUARE which uses more mathematical notation. In METU, various implementations of this sublanguage is handled (8).

In SEQUEL the retrieval operation is by using the SELECT statement. The general syntax is

SELECT [UNIQUE] attribute-list
FROM relation-list
[WHERE qualification]
[GROUP BY attribute-name [HAVING qualification]]

Here the attribute-list is the attributes which will be selected. The relation-list are the relations from which the attribute values are to be selected. The qualification in where statement is used to specify the conditions for selecting the tuples from one or more relations. GROUP BY groups the attributes of a relation according to a condition. UNIQUE must be specified if no duplication is allowed in the retrieval.

A simple retrieval query for getting COURSE-NAME attribute of all tuples of COURSE is :

SELECT COURSE-NAME
FROM COURSE.

This gives the relation COURSE projected on COURSE-NAME attribute.

Qualified retrieval operations perform the mapping operation using SELECT-FROM-WHERE block. For finding the COURSE-CODE in COURSE with the TYPE="DEPT" condition the following example can be given:

```
SELECT  COURSE-CODE
FROM    COURSE
WHERE   TYPE= 'DEPT'.
```

The qualification in WHERE clause can use the relational operators =, ¬=, >, ≥, <, ≤, and the Boolean operators NOT, AND, OR.

Nested levels of SELECT statement can be used in retrieval. So that the result of the innermost SELECT is used in qualification of another SELECT statement. This facility is used to express a join operation :

```
SELECT  SURNAME-NAME
FROM    STUDENT
WHERE   STUDENT-NO =
            SELECT  STUDENT-NO
            FROM    GRADES
            WHERE   COURSE-CODE = 'CS112'.
```

As the result of the above query, the names of the students taking the course CS112 are retrieved. First the numbers of the students taking the course are retrieved from GRADES; then these are matched those in STUDENT to obtain names.

Nesting of SELECT statements may be several levels and these specify a disjunctive mapping between an inner statement and the one that contains that.

Conjunctive mapping is for all elements of the list produced by the inner SELECT statement which is equivalent to a division operation, and specified by keywords GROUP BY, HAVING, SET.

The following query retrieves the student numbers of all students who has taken at least the courses taken by the one with student number 3665.

```
SELECT      STUDENT-NO
FROM        GRADES
GROUP BY    STUDENT-NO
HAVING SET( COURSE-CODE ) CONTAINS
```

```
SELECT   COURSE-CODE
FROM     GRADES
WHERE    STUDENT-NO = 3665 .
```

In SEQUEL, retrieval involving set comparisons can be done. For retrieving the courses that are offered and the names of the students taking them, the query is written as :

```
SELECT UNIQUE COURSE-CODE, SURNAME-NAME
FROM          GRADES, STUDENT
WHERE         GRADES.STUDENT-NO=STUDENT.STUDENT-NO .
```

SEQUEL data sublanguage has some built-in functions such as COUNT, SUM, AVG, MAX, and MIN. These help the user to perform such functions in retrieval operations.

SEQUEL also provides facilities for inserting, deleting, and updating tuples. The general syntax for these statements are:

```
INSERT INTO relation-name(attribute-list):
                                  <value-list> .
```

The values for the attributes to be inserted are given in value-list, if any attribute is not specified null value is assigned.

```
DELETE  relation-name
WHERE   qualification .
```

All the tuples that satisfy the condition will be deleted from the relation.

```
UPDATE relation-name
SET    attribute-name = update-value
WHERE  qualification .
```

In the update operation the tuples to be updated are selected and the new values are specified.

Examples : (1) Insert a record into the relation GRADES:

```
INSERT INTO GRADES ( STUDENT-NO,COURSE-CODE,GRADE):
          <3650,'CS222',FF> .
```

(2) Delete the course with the number 'CS222':

```
DELETE  COURSE
WHERE   COURSE-CODE = 'CS222'.
```

(3) Change the attribute GRADE in the relation GRADES :

```
UPDATE  GRADES
SET     GRADE = 'AA'
WHERE   STUDENT-NO = 3650
AND     COURSE-CODE = 'CS222' .
```

1.2.2 DL/1

DL/1 is the data sublanguage of the Information Management System (IMS) used in IBM systems (2, 4). It is a hierarchical data language allowing retrieval, insertion, deletion, and update of records in a hierarchical database. It is a host data language and is invoked from application programs written in PL/I, COBOL, and IBM Assembler Language by procedure calls. DL/1 uses preorder database tree traversal, starting at the root record visiting all the records in the database tree in left to right order (5).

In DL/1, retrieval is through use of the GET commands. GET UNIQUE (GU for short) is for direct retrieval of a particular record of a specified type. This command sets the current record pointer at a starting position for sequential processing. The selected record willbe the leftmost one; so the operation must be controlled via a condition.

Example : Retrieve the first record of STUDENT where FACULTY = 'ENG' :

```
GU  COURSE
    STUDENT ( FACULTY = 'ENG' )
```

Starting from the current position in the database tree, sequential retrieval can be achieved using the GET NEXT (GN for short) command. All records before the current position are eliminated. If a record type or a condition is not specified in a GN command, then the next record according to the preorder traversal is retrieved.

The retrieval may be restricted to records of one type by specifying a record name. For example, to retrieve the next STUDENT record, the query must be :

```
    GN  STUDENT
```

By specifying a condition, the records satisfying the condition will be selected in the sequential traversal of the database tree; for example :

```
    GN  STUDENT ( FACULTY = 'ENG' )
```

Example : A complete query for finding all the students in the ENG department is:

```
    GU  COURSE
        STUDENT ( FACULTY = 'ENG' )
NS  GN  STUDENT ( FACULTY = 'ENG' )
    go to NS
```

Here, 'go to NS' is a statement of the host language in which DL/1 is used.

Records within the family of a parent record are obtained ubing the command GET NEXT WITHIN PARENT (GNP). The parent record will be the one which is retrieved by using a GU or GN command. The retrieval ends when the last child of the parent record is fetched. The command may be used with or without a specified record type name and a qualification.

Example : To get all students who scored a grade of 'AA' in the course 'CS122' the query will be

```
    GU   COURSE ( COURSE-CODE = 'CS122' )
NS  GNP  STUDENT ( GRADE = 'AA' )
    go to NS
```

All the records linked to the parent node 'CS122' can be retrieved sequentially with the example query :

```
    GU  COURSE ( COURSE-CODE = 'CS122' )
NN  GNP
    go to NN
```

GET commands may also be used with the HOLD option (GHU, GHN, GHNP) which retrieves the record and allows for a subsequent modification operation.

INSERT (ISRT for short) command is used to add a new record to the database, and also to initially load it. Before insertion, the record which will be the parent of the new one must be selected to bo the current record. The INSERT command stores the record and creates its link to its parent.

Example : When a new student is registered to the course 'CS222', the query for insertion can be

```
ISRT  COURSE ( COURSE-CODE = 'CS222' )
      STUDENT
```

Also an insertion to the current position, which was accessed following a GET or INSERT command, is possible with an unqualified operation. For example,

```
ISRT  STUDENT
```

The command DELETE (DLET for short) deletes a record and all of its decendents. The record to be deleted must be retrieved using one of the GET HOLD commands.

Example : Delete the teacher who is giving the course 'CS122' :

```
GHU   COURSE ( COURSE-CODE = 'CS122' )
      PROFESSOR
DLET
```

The REPLACE (REPL for short) is used to update a record of the database. Again the record to be changed must be retrieved using the GET command with the HOLD option.

Example : Change the grade of the student attending the course 'CS122' and with student-number 3650 :

```
GHU   COURSE ( COURSE-CODE = 'CS122' )
      STUDENT ( STUDENT-NO = 3650 )
      change the value of GRADE to 'AA'
      in the workarea
REPL
```

1.2.3. DMS II

Database Management System (DMS II) is a Burroughs' sofware product (15,16,17). It is often used as a network database management system, but can be designed to follow one of the data models of hierarchical, network, or relational (13). The UNIVERSITY database is shown in network model with M:N relationship between COURSE and STUDENT in Figure 1.3.

The data language consists of a Data And Structure Definition Language (16) and a retrieval and manipulation language. DASDL is used by the database administrator to describe the "logical skeleton" of a database and the formats of the records it contains (Figure 1.4). The data retrieval and manipulation language of DMS II can be used from one of the host languages COBOL, PL/I, and ALGOL (15), or from the query language DMS INQUIRY (17), which may be used from a terminal.

This section summarizes the host data language commands.

The retrieval commands are FIND and MODIFY. Both are used to locate and fetch a record from a data set to a record area. The difference is that MODIFY statement locks the record to prevent another user from concurrent modification.

Example : FIND COURSE ON EXCEPTION GO XX.

In the above statement, the record being pointed to by the current record pointer is retrieved. 'ON EXCEPTION' clause must be included in all DMS II host language commands. The options NEXT, PRIOR, FIRST, LAST specify the position of the record in physical order.

Example : FIND NEXT COURSE ON EXCEPTION GO X.

A subset of the data set (similar to a conventional file) can be specified using the second kind of retrieval statements. Here the subset is formed by the keys for access to the data set. Same options as above apply here, too.

```
COURSE  DATA SET
( COURSE-CODE  ALPHA ( 7);
  COURSE-NAME  ALPHA (25);
  CREDIT       NUMBER( 2);
  TYPE         ALPHA ( 5);
  TAKEN BY SUBSET OF STUDENT KEY STUDENT-NO ;
  PROFESSOR DATA SET
  ( P-NUMBER   NUMBER( 5);
    PROF-NAME  ALPHA (25);
  );
  GIVEN-BY SET OF PROFESSOR KEY P-NUMBER ;
);
FIND-COURSE SET OF COURSE KEY COURSE-CODE
STUDENT DATA SET
( STUDENT-NO     NUMBER( 4);
  SURNAME-NAME   ALPHA (25);
  FACULTY        ALPHA ( 4);
  DEPT           ALPHA ( 4);
  YEAR           NUMBER( 1);
  GRADES         GROUP OCCURS 10 TIMES
                 ( GRADE     ALPHA ( 2) );
  COURSE-TAKEN SUBSET OF COURSE KEY COURSE-CODE;
);
FIND-STUDENT SET OF STUDENT KEY STUDENT-NO;
```

Figure 1.4 DASDL definition for the network database UNIVERSITY.

Example : FIND COURSE VIA FIND-COURSE AT COURSE-CODE= 'CS122' ON EXCEPTION GO Z.

The modification statements used in DMS II data language are : STORE, DELETE, INSERT, REMOVE, FREE, CREATE (RECREATE).

STORE statement is used to write a new record into a data set or to replace an existing one with a modified

copy. CREATE must be used to initialize a record area associated with a given data set before a new record can be stored in the data set. DELETE statement is used to find and delete a specific record from a data set; insertion of a record into a subset (creating a link) is possible by the INSERT command. A record in a subset may be removed (breaking a link) by the REMOVE statement. FREE statement unlocks the current record in a data set.

Examples :

(1) Create a new record in the course file:

```
CREATE COURSE ON EXCEPTION GO C-EXCEPTION.
MOVE NEW-CC TO COURSE-CODE.
MOVE NEW-CN TO COURSE-NAME.
MOVE NEW-C  TO CREDIT.
MOVE NEW-T  TO TYPE.
STORE  COURSE ON EXCEPTION GO C-EXCEPTION.
```

(2) Change the name of the course 'CS112' :

```
FIND COURSE VIA FIND-COURSE AT COURSE-CODE='CS112'
                              ON EXCEPTION GO X.
MOVE 'INTR. TO COMPUTATION' TO COURSE-NAME.
STORE COURSE ON EXCEPTION GO X.
```

(3) Drop the last student attending 'CS112' :

```
MODIFY COURSE VIA FIND-COURSE AT COURSE-CODE =
                'CS112' ON EXCEPTION GO X.
MODIFY STUDENT VIA LAST TAKEN-BY ON EXCEPTION GO X.
REMOVE CURRENT FROM COURSE-TAKEN ON EXCEPTION GO X.
REMOVE CURRENT FROM TAKEN-BY ON EXCEPTION GO X.
```

(4) Delete the student whose number is 3660 :

```
FIND STUDENT VIA FNID-STUDENT AT STUDENT-NO =
          3660 ON EXCEPTION GO T.
REMOVE-LINKS.
FIND COURSE VIA NEXT COURSE-TAKEN
          ON EXCEPTION GO DELETE-RECORD.
REMOVE CURRENT TAKEN-BY ON EXCEPTION GO T.
```

```
    REMOVE CURRENT COURSE-TAKEN ON EXCEPTION GO T.
    GO    REMOVE-LINKS.
DELETE-RECORD.
    DELETE STUDENT ON EXCEPTION GO T.
```

(5) Find all the students attending 'CS222' :

```
    FIND COURSE VIA FIND-COURSE AT COURSE-CODE='CS222'
                                ON EXCEPTION GO T.
NEXT-RECORD.
    FIND STUDENT VIA NEXT TAKEN-BY ON EXCEPTION GO T.
    GO    NEXT-RECORD.
```

(6) Find the professor giving the course 'CS222' :

```
    FIND COURSE VIA FIND-COURSE AT COURSE-CODE = 'CS222'
                                ON EXCEPTION GO T.
    FIND PROFESSOR VIA FIRST GIVEN-BY
                                ON EXCEPTION GO T.
```

1.2.4 DDS Data Language

DDS data language is the data language of the Dynamic Database Management System (7). It is a high-level, nonprocedural, and relationally complete language; the commands are in Turkish. It is composed of three sublanguages: data definition, data manipulation, and data retrieval sublanguages.

The data definition sublanguage (10) consists of two main groups of definition commands, one for database definition and the other for modification of definitions in the database system. The user is allowed to define the database step by step, and the database may be used only after the definition of all the components are complete. The statements of the data definition sublanguage is'VERİ TEMELİ TANIMI','KÜTÜK İÇERİĞİ', 'GİBİDİR', 'KALICIDIR', 'GEÇİCİDİR'.

DDS data manipulation sublanguage (12) includes the statements 'YARAT', 'AKTAR', 'BİRLEŞTİR', 'EKLE', 'SİL', 'SIRALA', 'GÜNLE', 'DEVİR'. The functions of the language are to maintain and update a database and to

define efficient access paths. The topic on the access paths are taken in detail in Section 2.4.

The data retrieval sublanguage of DDS is made up of the varieties of the basic 'ÇEK' statement. There are four types: Unconditional, keyed, conditional, and set conditional statements. These are explained in detail in Chapter 3.

1.3 Purpose And The Scope Of The Study

The purpose of this work is to design and implement a software system for the data retrieval sublanguage for the DBMS called Dynamic Database Management System (DDS). This is part of a series of studies conducted to fully implement DDS. The data definition sublanguage of DDS has already been implemented (10). Together with the completion of the translator (11), and the data manipulation sublanguage (12), the implementation of DDS will be completed.

The need for the DBMS is to handle unanticipated queries of various users. The objective of the retrieval part is to do this with fast and flexible access capability.

In the thesis after the introduction, Chapter 2 is devoted to a general presentation of the dynamic database management system DDS. After giving a survey of the basic features of DDS, and the architectural aspects of DDS, the chapter concludes with a review of the capabilities of the DDS data language. This review also sets the grounds for a detailed presentation of the data retrieval facilities of DDS, in the next chapter.

Chapter 3 is a presentation of the data retrieval sublanguage of DDS. The syntax and semantics of the data retrieval statements of the sublanguage as designed by Uçkan (7) are explained in this chapter.

The development of the software and application examples are taken up in Chapter 4. After an analysis of the data retrieval facilities of Cobol and explanation of the structure of the software system, the individual

define efficient access paths. The topic on the access paths are taken in detail in Section 2.4.

The data retrieval sublanguage of DDS is made up of the varieties of the basic 'ÇEK' statement. There are four types: Unconditional, keyed, conditional, and set conditional statements. These are explained in detail in Chapter 3.

1.3 Purpóse And The Scope Of The Study

The purpose of this work is to design and implement a software system for the data retrieval sublanguage for the DBMS called Dynamic Database Management System (DDS). This is part of a series of studies conducted to fully implement DDS. The data definition sublanguage of DDS has already been implemented (10). Together with the completion of the translator (11), and the data manipulation sublanguage (12), the implementation of DDS will be completed.

The need for the DBMS is to handle unanticipated queries of various users. The objective of the retrieval part is to do this with fast and flexible access capability.

In the thesis after the introduction, Chapter 2 is devoted to a general presentation of the dynamic database management system DDS. After giving a survey of the basic features of DDS, and the architectural aspects of DDS, the chapter concludes with a review of the capabilities of the DDS data language. This review also sets the grounds for a detailed presentation of the data retrieval facilities of DDS, in the next chapter.

Chapter 3 is a presentation of the data retrieval sublanguage of DDS. The syntax and semantics of the data retrieval statements of the sublanguage as designed by Uçkan (7) are explained in this chapter.

The development of the software and application examples are taken up in Chapter 4. After an analysis of the data retrieval facilities of Cobol and explanation of the structure of the software system, the individual

statements are covered. The data retrieval sublanguage statements are explained as to their algorithms and implementation issues.

Chapter 5 summarizes the thesis and its findings.

This is followed by the list of references and appendices which contain :

-Lexical tokens of DRS processor,
-Sample runs,
-Program listings of the software developed.

2. DYNAMIC DATABASE MANAGEMENT SYSTEM (DDS)

DDS which is based on the relational model has its data language based on the natural structure of Turkish. In this section the general structure of the DBMS called Dynamic Database Management System (DDS), and its capabilities will be introduced.

2.1. Basic Features of DDS

Basic features of DDS designed by Uçkan (7) are as follows:

1. DDS is based on the relational model.

2. The database of the system is integrated, so that it can be used by more than one application and the problem of redundancy is eliminated.

3. System security can be achieved in data item level as well as in database level. Each user has a security level assigned which limits the extent of operations that can possibly be carried by him, and also his access to the database.

4. There will be a system file providing system use statistics. The database adminitrator can get information on system security matters.

5. The structural, syntactical and semantical errors made by the users will be kept in a system file and the users will be informed of them.

6. The data language of DDS is based on the natural structure of Turkish language, and it is composed of data definition, data manipulation, and data retrieval sublanguages.

7. DDS data definition sublanguage has the self defining capability. The whole database can be defined by the use of it. The database has two types of files: user files which store attribute values for various entities, and the system files which store the necessary information about both the system and the user databases.

8. The user may define the whole database at once or in parts at different times. The latter approach will prohibit the use of the database until it is completely defined.

9. The system has the capabilities of defining identical files and of deleting definitions of files.

10. The system can give the user a subset of the database definition, thus the DDS may function as a restricted Database Administrator.

11. DDS data manipulation sublanguage has the capability of performing all of the necessary operations for maintaining and updating the database.

12. DDS data retrieval sublanguage is relationally complete.

13. Besides the above capabilities, the DDS data language can perform calculations on the retrieved data, decode the output using the data dictionary, and report the results.

14. Although the DDS data language is based on the relational model, the relational terminology is not used; data processing terminology which is simpler and natural for the user is chosen.

15. DDS data language is block structured so that each operation is defined in a block. Although it is designed as a stand-alone language, it may also be used in a host language such as COBOL, PL/I, ALGOL, etc.

16. DDS database management system can operate in batch or interactive mode, but batch mode is emphasized in the design of the system.

17. DDS has the capability of external/conceptual and conceptual/internal map handling (2).

18. The file organization methods used are direct access for user files, and index-sequential access for the system files. The user must define the hash functions for accessing his files and these can be changed by use of the DDS data language.

19. For retrieval on secondary keys DDS has an

effective access strategy. Before the retrieval operation, DDS inverts the file on the given keys and create an access path. This access path will be saved in a system file for possible future use. The seldom used access path entries will be automatically deleted so that newly created ones can be entered.

20. Inversion operation can be explicitly done on secondary keys, upon user request. This implies inversion before data retrieval operations.

21. Sorted sequence of user files on certain data items may be kept in a system file without effecting the physical organization of the file. The sorting strategy is very similar to that of inversion.

2.2 Architecture And Elements of DDS

DDS architecture (7) is given in Figure 2.1 and can be grouped as :

1. Algorithmic components: These build up the system software and are specified in detail in (7). The components concerning the retrieval part are given in Section 2.4.

2. Secondary system components : These are the basic data structures used by the software, and are called system files. They contain all the information necessary for various operations of the DDS system and are kept in auxiliary storage.

The system files in DDS architecture are listed according their functions.

a) The Security Control System:

(i) SISK01- User Security Status File:

It keeps the user number, name, and security code assigned by the DBA. Only the users defined in this file can access the database; and their security code restricts the access to data items and operations.

(ii) SISK04- Record Definition File:

The type, length, security code, and other attributes of the data items are kept in this file. The system security at data level is maintained by the use of

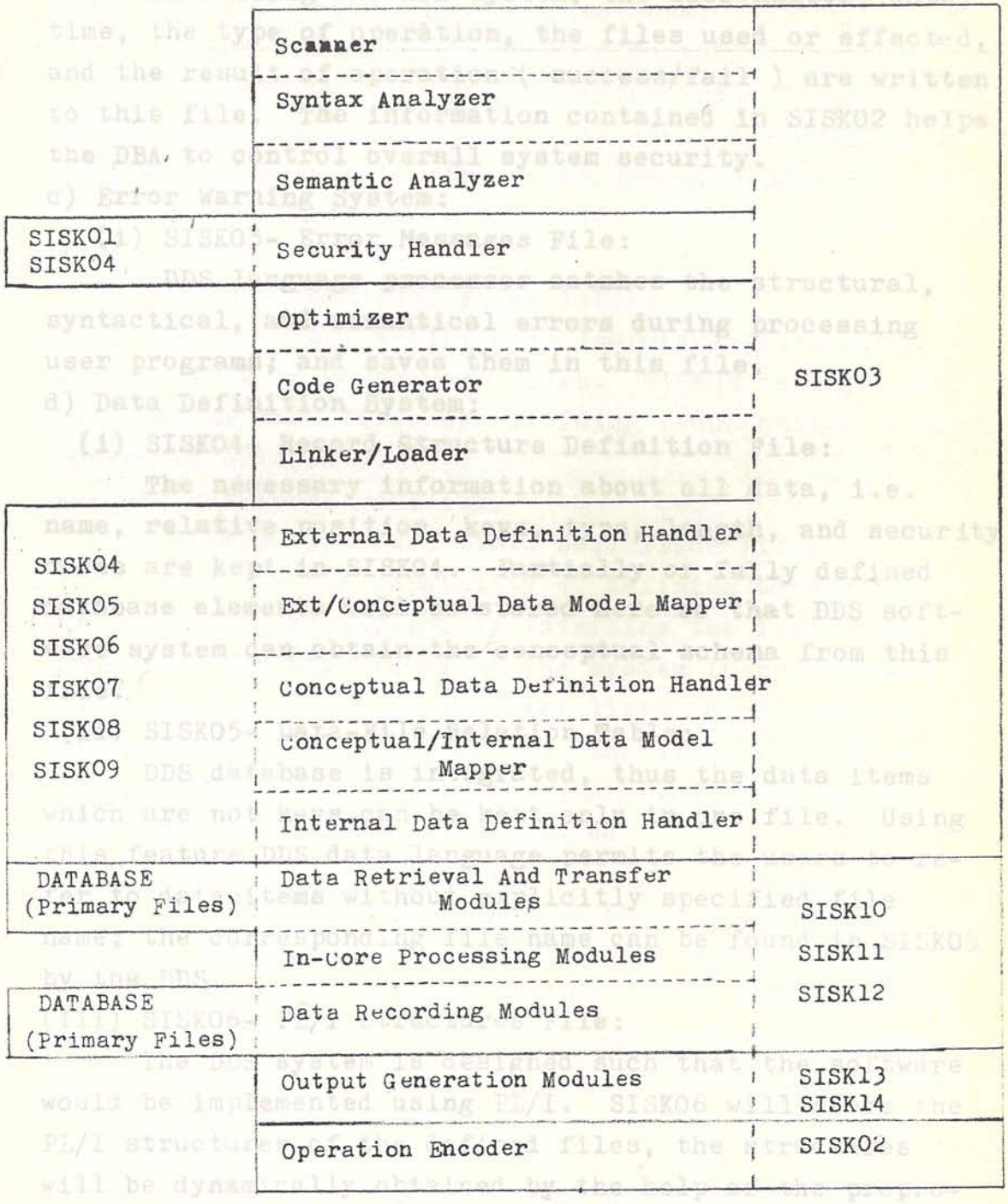

Figure 2.1 DDS Database Management System Architecture.

this file.

b) Journalling System:

(i) SISK02- Journal File:

When using the DDS system, the user number, date, time, the type of operation, the files used or effected, and the result of operation (success/fail) are written to this file. The information contained in SISK02 helps the DBA to control overall system security.

c) Error Warning System:

(i) SISK03- Error Messages File:

DDS language processor catches the structural, syntactical, and semantical errors during processing user programs; and saves them in this file.

d) Data Definition System:

(i) SISK04- Record Structure Definition File:

The necessary information about all data, i.e. name, relative position, keys, type, length, and security codes are kept in SISK04. Partially or fully defined database elements will be stored here so that DDS software system can obtain the conceptual schema from this file.

(ii) SISK05- Data-File Relation Table:

DDS database is integrated, thus the data items which are not keys can be kept only in one file. Using this feature DDS data language permits the users to refer to data items without explicitly specified file name; the corresponding file name can be found in SISK05 by the DDS.

(iii) SISK06- PL/I Structures File:

The DDS system is designed such that the software would be implemented using PL/I. SISK06 will store the PL/I structures of the defined files, the structures will be dynamically obtained by the help of the preprocessor in PL/I compilers and % INCLUDE statement (18).

(iv) SISK07- Hash Function Guide:

The user files in DDS will be internally organized using some key-to-address transformation methods. The

user can define an efficient hash function to the system or change the internal organization of a file by changing its hash function. This file keeps the names of the user files and their hashing functions.

(v) SISK08- Database Definition Control File:

DDS permits the user to define the files in steps. SISK08 controls whether the files are fully defined or not; and it does not permit the use of a partially defined file. Furthermore it avoids duplicate definitions while adding new ones.

(vi) SISK09- Identical Files Table:

The user may define some temporary files similar in structure to other user files. SISK09 is employed in defining such files and avoiding redundancies without paying in integrity.

e) Data Retrieval Facility:

(i) SISK10- Inverted Lists Definition File:

DDS has the capability of defining access paths by explicitly or implicitly inverting the files. This increases the efficiency of the system in retrieval operations. Besides the inverted lists, additional information about the amount of time the lists will be kept in the system will be stored in SISK10. This system file will be automatically updated.

(ii) SISK11- Key Lists File:

This file contains key lists of every file in the system. SISK10 contains the inverted lists as a bit string whose size is equal to the record number in the file. The access to the user files is directly through keys. The mapping of the strings in SISK10 to the keys in SISK11 permits access directly to the user files.

(iii) SISK12- Sort File:

DDS is able to sort files according to data items that are not keys without changing the physical organization of the file. The relative positions of records in sorted order are kept as linear lists in SISK12. Similar to inversion, mapping of the linear lists to the

keys in SISK11 permits access directly to the user file in sorted order.

f) Data Encoding/Decoding System:

(i) SISK13- Data Transformation System Relation Table:

The data retrieved during DDS operations are kept in user's workspace and then transmitted to the output media. The DDS system may decode the data using SISK13 and SISK14. SISK13 contains all the attributes in the system with correponding transformation system if it is coded.

(ii) SISK14- Data Transformation File:

This file contains the coded attribute values with corresponding decoded forms.

2.3. DDS Data Language

DDS data language is a nonprocedural high-level relational language consisting of 24 statements. It consists of three sublanguages:

1. Data Definition Sublanguage,
2. Data Manipulation Sublanguage,
3. Data Retrieval Sublanguage.

This section will contain the summary of DDS data language statements and capabilities.

a) Data Definition Sublanguage Statements:

Data definition sublanguage permits the user to define the database either as a whole or partially in steps. Also, defining files that are identical to those that already exist in the database or deleting such definitions is possible.

(i) 'VERİ TEMELİ TANIMI' : It is a block of statements used for defining the components of the database.

(ii) 'KÜTÜK İÇERİĞİ': This also is a block of statements where the user defines the contents of a file specified in a DDS program.

(iii) 'GİBİDİR': Defines two files that are identical in record structure.

(iv) 'KALICIDIR': The user file definitions will be

made permanent.

(v) 'GEÇİCİDİR': The file definitions will be deleted.

b) Data Manipulation Sublanguage Statements:

The object of the DDS data manipulation sublanguage is to maintain and support an up-to-date database, and to define efficent access paths.

(i) 'YARAT': Relates the definition of a file to its content, i.e. creates it.

(ii) 'AKTAR': Copies the file or divides it into a few.

(iii) 'BİRLEŞTİR': Merges all or some of the contents of two files.

(iv) 'EKLE': Add new records to the file.

(v) 'SİL': Used to delete records from the file.

(vi) 'SIRALA': Sorts the file according to specified sorting keys.

(vii) 'GÜNLE': Changes the contents of records in the file.

(viii) 'DEVİR': Explicitly invert the file and define the access paths. The user can actually invert the file using this statement before retrieval.

c) Data Retrieval Sublanguage Statements:

DDS data retrieval sublanguage is used to find answer to the queries of the various users, and places the contents of the retrieved data attributes to the workarea. It consists of four retrieve statements, capable of performing all the operations defined in relational algebra. Unconditional retrieve corresponds to the projection operation and conditional retrieve corresponds to the selection operation in relational algebra. Expressing all of the operations relational algebra by DDS data language statements, the relational completeness of the system is proved (7). The syntax and semantics of these statements will be explained in detail in Chapter 3.

(i) Unconditional retrieve statement: All the records of the file is accessed and the contents of all data items are retrieved.

(ii) Keyed retrieve statement : Retrieve data using specified key fields.

(iii) Conditional retrieve statement: Retrieve all data satisfying a given condition.

(iv) Set conditional retrieve statement: Retrieve the data which satisfies the given set condition.

d) Auxiliary Statements:

These statements are used after the retrieval statements. They do some additional operations on the contents of the workarea.

(i) 'SAY': Counts the attribute values of the files that are retrieved.

(ii) 'BUL': Used to find the sum, maximum, minumum, or average of the data values of the files in the workarea.

(iii) 'YAP': A name is assigned to the work file.

(iv) 'SAKLA': The contents of the workarea are marked as to be saved.

(v) 'DÖNÜŞTÜR': Decodes the information in the workarea.

(vi) 'YAZ': The contents of the workarea will be transferred to the output media.

In addition to these, comments may be inserted in to the source program for documentation purposes.

2.4. DDS Dynamic Data Retrieval Facility

An important aim of the DDS is to facilitate data retrieval operations. Retrieving the elements of a database efficiently is generally an important problem. Various file organization techniques are developped for this purpose (6). One of the methods is concerned with the use of so called 'access paths' (4). The access paths are defined as the physical connection between data which implement a relationship or a search-aiding data structure. The DBMS uses access paths for selecting the appropriate data for a particular request. An inverted file may be used to provide the access path corresponding to the condition. However, keeping these in

memory results in waist, saving can be effected by keeping the access paths as bit arrays. In DDS, The file is inverted according to a data attribute when it is explicitly wanted by the user or need arises in a retrieval statement. Thus defined path is kept in one of the system files.

After inverting a file according to a data, there will be an inverted list so that every element corresponds to the value of the data. As an example, inverting the file COURSE according to TYPE creates two inverted lists: one for TYPE= 'DEPT' which will be

(CS112, CS122, CS214, CS222)

and the other for TYPE= 'GEN' which will be

(CS252, ENG104, MATH152, PHYS106).

This is called inversion according to a data, or 'complete inversion'. Also, by inverting a file according to data satisfying a specific condition The so called 'partial inversion' is obtained. The latter results in a subset of inverted lists for that file.

To keep the inverted lists as a list of keys will not be economical in large files. It is possible to have a bit string as long as the number of records in the file. The bit corresponding to a record satisfying the condition will be made '1'B, otherwise '0'B. The bit string so formed is called the access path.

Access path belongs to an attribute-value pair. This means that, a condition as

(TYPE = 'GEN') VEYA (CREDIT $\geqslant$ 12)

creates three access paths:

AP(TYPE = 'GEN') = 00001111

AP(CREDIT = 12) = 00001110

AP(CREDIT = 15) = 00000001

To satisfy the condition given above all three access paths thus found are connected via an OR operation:

AP((TYPE='GEN')VEYA(CREDIT=12)VEYA(CREDIT=15)= 00001111.

This is the access path that is used in accessing the records whose type is general or credit is 12 or 15.

The last four records in the file satisfy these conditions.

The efficient and dynamic usage of the access path method requires two system files: SISKlO (inverted lists definition file) and SISKll (key list file) (see section 2.2).

Each record of SISKlO contains an access path. Another attribute of the file is the inversion-type which may be 'TAM' if the file is inverted according to data, 'KISMİ' if it is inverted because of a condition. The creation-type can contain the value 'DEVİRME' if created as a result of an invert statement, 'ÇEKME' if created after a conditional retrieval statement. Two attributes, usage-count and last-access-date, are kept for deleting least used entries from the file whenever necessary.

The second file, SISKll, keeps the list of keys used to access the user file. The access to the user files will be direct via a key. It was explained above that using a key for each access path in SISKlO is not economical. Hence, the keys to all user files will be kept in SISKll, and mapping the access path to the keys will give the keys to access the file.

This dynamic access method using SISKlO and SISKll is explained below.

(1) The inversion of a file according to a data or condition and creating the access path is possible, by explicitly using the DDS data manipulation sublanguage statement 'DEVİR'. If an access path already exists in SISKlO, it is used and the last-access-date of it is updated. Otherwise, the related records are created.

(2) The conditional retrieval statement acts as an implicit inversion operation, and causes partial inversion. The access paths thus obtained are appended to SISKlO.

(3) Upon a data retrieval query, the system checks whether the required access path exists or not. If necessary, the access paths are created and written to

SISK10 and by mapping these to the keys in SISK11 the data is retrieved. If the access paths are already defined, only the usage-count and the last-access-date of the records are updated.

(4) The size of the system file SISK10 is constant. Inversion and conditional data retrieval queries to the DDS system will increase the number of access paths, and as a result, the file will become full. At this point, in order to add new access paths, some of the records have to be deleted. The replacement policy is to pick the record with the oldest last-access-time entry. If there are more than one record with equal oldness, then the one with the smallest usage-count entry gets deleted.

3. DATA RETRIEVAL FACILITIES OF DDS

Data retrieval sublanguage of DDS consists of four kinds of retrieve statements, and some auxiliary statements. In the following sections the syntax and semantics of these statements will be explained (7).

3.1. The Notation

The syntax of the sublanguage is described in CBL (COBOL-like). Each production in the notation has the form :

$\langle S \rangle$ ::= E

Where S is a nonterminal and E is an expression composed of nonterminals and terminals denoting the substitution for S.

An expression E may consist of several items in the form :

$e_1\ e_2\ e_3\ \ldots$

Where each e_i may be,

i . a terminal symbol

ÇEK

ii . a nonterminal symbol

<file name>

iii . a sequence of e_i's enclosed in square brackets, indicating an option

[<file name> KÜTÜĞÜNDEN]

iv . two or more sequences of e_i's enclosed in braces indicating a selection

{VERİSİNİ / VERİLERİNİ}

v . two or more sequences of e_i's enclosed in square brackets indicating an optional selection

[<digit> / <letter>]

(...) represents the position at which repetition may occur at the user's option. All underlined upper-case words are the terminals.

3.2. The Lexical Elements

a) Character set :

Consists of letters, digits, and special characters.

$$\langle \text{letter} \rangle ::= \{ A \ldots Z \}$$

$$\langle \text{digit} \rangle ::= \{ 0 \ldots 9 \}$$

b) Identifiers :

$$\langle \text{name} \rangle ::= \langle \text{letter} \rangle \left[\begin{matrix} \langle \text{digit} \rangle \\ \langle \text{letter} \rangle \\ \_ \end{matrix} \right]$$

There are some qualified names which will be mentioned in the text, as ⟨file name⟩, ⟨data name⟩, ⟨program name⟩, etc.

c) Constants :

$$\langle \text{data value} \rangle ::= \{ \langle \text{number} \rangle \; \langle \text{string} \rangle \}$$

$$\langle \text{number} \rangle ::= \langle \text{digit} \rangle \; [\langle \text{digit} \rangle] \ldots$$

$$\langle \text{string} \rangle ::= \{ \emptyset \; \langle \text{digit} \rangle \; \langle \text{letter} \rangle \} \left[\begin{matrix} \emptyset \\ \langle \text{digit} \rangle \\ \langle \text{letter} \rangle \end{matrix} \right] \ldots$$

d) Comments :

$$\langle \text{comment} \rangle ::= /* \; \langle \text{string} \rangle \; */$$

3.3. Structure of DDS Data Retrieval Programs

DDS Data retrieval program can consist of one or several data retrieval statements and auxiliary statements.

⟨DDS data retrieval program⟩ ::= [⟨program name⟩ :]

BAŞLA : {⟨S1⟩ / ⟨S2⟩ / ⟨S3⟩} [[⟨S3⟩] ⟨S4⟩] ... [[⟨S4⟩] ⟨S3⟩] ...

[{⟨count statement⟩ / ⟨find statement⟩}] [⟨convert statement⟩]

⟨write statement⟩] BİTİR.

Where :

⟨S1⟩ ::= ⟨unconditional retrieval statement⟩
[⟨change statement⟩] [⟨Save statement⟩]

⟨S2⟩ ::= ⟨conditional retrieval⟩
[⟨change statement⟩] [⟨Save statement⟩]

⟨S3⟩ ::= ⟨keyed statement⟩
[⟨change statement⟩] [⟨Save statement⟩]

⟨S4⟩ ::= ⟨Set conditional statement⟩
[⟨change statement⟩] [⟨Save statement⟩]

The DDS data retrieval program must start with the keyword 'BAŞLA' and end with 'BİTİR'. The first statement of the program must be any one of unconditional, conditional or keyed retrieval statements.

The definition of DDS data retrieval program shows that theoretically infinite structures can be produced. However, it is limited at least with the number of files in the database. This feature of the language shows that the query language is quite powerful, so that complex queries for, retrieval of data can be written.

3.4 Unconditional Retrieval Statement

Unconditional retrieval ('Koşulsuz ÇEK') is a simple retrieval statement. As a result of this statement the specified data items are retrieved from all of the records in the user file.

⟨Unconditional retrieval statement⟩ ::=

⟨file name⟩ KÜTÜĞÜNDEN TÜM VERİLERİ

[⟨file name⟩ KÜTÜĞÜNDEN] ⟨data name list⟩ { VERİSİNİ / VERİLERİNİ }

[⟨data name list⟩ ANAHTARINA GÖRE SIRALI] ÇEK.

⟨data name list⟩ ::= ⟨data name⟩ [, ⟨data name⟩] ...

Example : Find the names of all students in the file STUDENT.

STUDENT KÜTÜĞÜNDEN SURNAME-NAME VERİSİNİ ÇEK.

If 'TÜM' option is specified, then the whole record is retrieved.

Example : STUDENT KÜTÜĞÜNDEN TÜM VERİLERİ ÇEK.

Optionally, the retrieved data items may be sorted according to specified fields provided that they are not keys.

Example : SURNAME-NAME,FACULTY VERİLERİNİ FACULTY ANAHTARINA GÖRE SIRALI ÇEK.

In all retrieval statements it is optional to specify the file name in the statement. If it is not specified, then the file name is determined to be the one common to all data names. If the data names specified in the data name list are keys or 'TÜM' option is specified and hence all of the data in the file is to be retrieved then the file name must be explicitly specified in the statement.

As a result of a retrieval statement a workfile called 'ÇIKTIKÜT' is created. In a subsequent retrieve statement this workfile may be used as a file from which data items may be retrieved.

3.5. Keyed Retrieval Statement

The keyed retrieval statement ('Anahtarlı ÇEK') causes quick access to the records with specified keys. As a result of the execution of this statement the

contents of data items specified in the data name list will be placed into the workarea.

⟨keyed retrieval statement⟩ ::=

{ ⟨file name⟩ KÜTÜĞÜNDEN ⟨key list⟩ ANAHTARLI
{ [⟨file name⟩ KÜTÜĞÜNDEN] ⟨key list⟩ ANAHTARLI

TÜM VERİLERİ
⟨data name list⟩ { VERİSİNİ / VERİLERİNİ } } ÇEK.

⟨key list⟩ ::= { ⟨open key list⟩ [,(⟨set name⟩)] / [(⟨set name⟩),] [⟨open key list⟩] }

⟨open key list⟩ ::= { ⟨key⟩ / ⟨key⟩ - ⟨key⟩ } [⟨key⟩ / ⟨key⟩ - ⟨key⟩] ...

⟨key⟩ ::= ⟨data value⟩

In the DDS system, the user files are directly accessed ; and keys are part of the records. In order to find the relative location of a record in the file, the hash function of the user file are applied to the keys.

Using the keyed retrieval statement, the user can access the records with keys in a specific range, from key-1 upto key-2. The 'open key list' should be used to indicate such key ranges and key values.

Example : COURSE KÜTÜĞÜNDEN CS112,CS222-CS254 ANAHTARLI
TÜM VERİLERİ ÇEK.

As a result of a retrieval statement there may be a set of key values in the workarea. These key values can then be used in a subsequent keyed retrieval statement ; the user must specify this set of keys in the 'set name', between paranthesis.

Example : STUDENT KÜTÜĞÜNDEN (STUDENT-NO) ANAHTARLI
SURNAME-NAME VERİSİNİ ÇEK.

3.6. Conditional Retrieval Statement

The third type of retrieval statement is for conditional retrieval. This statement ('Koşullu ÇEK') selects those records satisfying the specified conditions, retrieves the contents of data items and places them to the workarea.

<conditional retrieval statement> ::=

{ <file name> KÜTÜĞÜNDEN <condition> KOŞULLU
{[<file name> KÜTÜĞÜNDEN] <condition> KOŞULLU

TÜM VERİLERİ
<data name list> { VERİSİNİ / VERİLERİNİ } } ÇEK.

<condition> ::= { <attribute-value couple>
(<condition>) VE / VEYA (<condition>)
(<condition>) OLMAYAN }

<attribute-value couple> ::= <data name> { = / ≠ / < / > / ≤ / ≥ } <data value>

Simple conditions are formed by using the relational operators. For example, YEAR ⩾ 2 is a simple condition. However, more complex conditions can be formed by joining simple conditions with the logical operators 'and' ('VE'), 'or' ('VEYA') and 'not' ('OLMAYAN'). An example of a complex condition may be

((YEAR ⩾ 2) VE (DEPT = 'MAN')) OLMAYAN

Example : Find the numbers of the students in the CS department and year greater than 1.

STUDENT KÜTÜĞÜNDEN ((DEPT = 'CS') VE (YEAR > 1)
KOŞULLU STUDENT-NO VERİSİNİ ÇEK.

Example : Find all about the given general courses.

COURSE KÜTÜĞÜNDEN TYPE = 'GENERAL' KOŞULLU TÜM VERİLERİ ÇEK.

3.7. Set Conditional Retrieval Statement

The syntax of the set conditional retrieval ('Küme koşullu ÇEK') statement is

<set conditional retrieval statement> ::=

<file name> KÜTÜĞÜNDEN { <set condition couple> KOŞULLU | (<set name>) DIŞINDAKİ }

[<file name> KÜTÜĞÜNDEN] { (<set condition couple>) KOŞULLU | (<set name>) DIŞINDAKİ }

TÜM VERİLERİ

<data name list> { VERİSİNİ | VERİLERİNİ } ÇEK.

<set condition couple> ::= <set name> { EŞİT | İÇERİR } (<set name>)

The values of data items in the workarea form a set, this set is compared with the set that is formed having the same key values in the user file, with some key operations.

'Set condition couple' is formed by combining a 'data name' with a name of a set in the workarea using the relations of set equality ('EŞİT') and set inclusion ('İÇERİR') operations. The 'set name' must be included in paranthesis. Another relation which can be defined on a set name is the set exclusion ('DIŞINDAKİ') relation which results in the retrieval of the data items for records which are not included in the set elements of the workarea.

Examples :

(1). GRADES KÜTÜĞÜNDEN (STUDENT-NO) DIŞINDAKİ
STUDENT-NO VERİSİNİ ÇEK.

(2). GRADES KÜTÜĞÜNDEN COURSE-CODE İÇERİR (COURSE-CODE)
KOŞULLU STUDENT-NO VERİSİNİ ÇEK.

3.8. Auxiliary Statements

Besides the retrieval statements, there are several other types of statements in the DDS Data Retrieval Language. These are the name 'auxiliary statements' by Uçkan (7). These are the change, save, convert, count, find and write statements.

a) The change statement ('YAP') may be used to change the name of the system workfile to a user defined name.

⟨change statement⟩ ::= ÇIKTI KÜTÜĞÜ ADINI ⟨file name⟩YAP.

b) The save statement ('SAKLA') is used to save from erasure the previously retrieved data items in the workarea. Data items thus saved are associated with the new retrived data by the next statement.

⟨save statement⟩ ::=

$$\left\{ \begin{array}{l} \text{TÜM VERİLERİ} \\ \langle\text{data name list}\rangle \left\{ \begin{array}{l} \text{VERİSİNİ} \\ \text{VERİLERİNİ} \end{array} \right\} \end{array} \right\} \text{SAKLA}$$

c) The count statement ('SAY') is used to count the values of some data items depending on a data name in the worarea.

$$\langle\text{count statement}\rangle ::= \left[\langle\text{file name}\rangle \text{ ÇIKTI KÜTÜĞÜNDEKİ}\right] \left[\text{HER } \langle\text{data name}\rangle \text{ VERİSİ DEĞERİ İÇİN}\right] \langle\text{data name list}\rangle \left\{ \begin{array}{l} \text{VERİSİ} \\ \text{VERİLERİ} \end{array} \right\} \text{ DEĞERLERİNİ SAY.}$$

d) The find statement ('BUL') is used to find the maximum, minimum, total, and average of the data values for each specific data attribute in the workarea.

$$\langle\text{find statement}\rangle ::= \left[\langle\text{file name}\rangle \text{ ÇIKTI KÜTÜĞÜNDEKİ}\right] \left[\text{HER } \langle\text{data name}\rangle \text{ VERİSİ DEĞERİ İÇİN}\right] \langle\text{data name list}\rangle \left\{ \begin{array}{l} \text{VERİSİ} \\ \text{VERİLERİ} \end{array} \right\} \text{ DEĞERLERİNİN} \left\{ \begin{array}{l} \text{TOPLAMINI} \\ \text{EN BÜYÜĞÜNÜ} \\ \text{EN KÜÇÜĞÜNÜ} \\ \text{ORTALAMASINI} \end{array} \right\} \text{ BUL.}$$

e) The convert statement ('DÖNÜŞTÜR') decodes the information in the workarea.

⟨convert statement⟩ ::= DÖNÜŞTÜR.

f) The write statement ('YAZ') is used to print the contents of a workarea.

⟨write statement⟩ ::= [ÇIKTI KÜTÜĞÜNÜ] YAZ.

4. SOFTWARE DEVELOPMENT

The Data Retrieval Sublanguage (DRS) of the DDS was implemented on Burroughs Medium System Computers (B3700, B2800, B2900). The programming language used in the implementation was COBOL.

Uçkan (7) had designed DDS with the thought of a PL/I implementation. However, the DRS implementation had to be done on a Burroughs Medium System Computer. Among the high level language compilers available in Medium Systems, COBOL has several extended capabilities in file organization that are extensions to the ANSI-74 COBOL. Thus for an efficient implementation COBOL was preferred.

In this chapter, first the general file organization methods supported by Burrougs COBOL Compilers and the ones used in the implementation are briefly explained. This is followed by an explanation of the components of the data retrieval subsystem and the implementation of each retrieval statement.

4.1. COBOL And Its Data Retrieval Facilities

There are three file organization methods in COBOL:

1. Sequential Organization
2. Relative Organization
3. Indexed Organization

In sequential file organization, the records can be accessed in the order they are written to the file. If a sequential file is used as input/output then a record can be read, updated and written to the previous position. The sequential I/O module provides the capability to access all of the records in the file in established sequence. The established sequence is the predecessor-successor relationship between the records, which do not change until records are added at the end of the file. Also the sharing of memory between the files are possible.

A relative organized file consist of records which are identified by relative record numbers. Relative record numbers are greater than zero and specify logical positions in the file. The storage and retrieval of each record is via this record number.

Access to unused or deleted record positions are inhibited. Each logical block containes additional information which are used by a set of compiler-provided intrinsics bound into user programs which declare a relative file.

A file created using indexed organization, consist of records that are uniquely identified by the value of one or more keys within the record. Each key is associated with an index which provides a logical path to the record according to the contents ofthe record key. The record key is used to identify the specific record for update, insert, and delete operations.

There are three access methods to retrieve information from a file. In SEQUENTIAL access, records may only be accessed sequentially. RANDOM access is for direct access to the record using a key. DYNAMIC access provides both sequential and random access to the file.

In sequential files only SEQUENTIAL and RANDOM access is provided. Relative and indexed files can be accessed using one of the three methods. When using RANDOM or DYNAMIC access method, the key will be the relative record number in relative files and the record key in indexed files.

In this study, the system files are indexed and the user files are sequentially organized. DYNAMIC access is used to system files, because in most files after direct access to the specific record, sequential reading is required. In sequential organized user files RANDOM access to the records is used. Transforming the key part using one of the hashing algorithms, gives the relative position of the record in the file.

One of the properties of COBOL used is the facility of referring different files by a general name in the program. The name used in the program is the 'internal file name' and 'external file name' is the name known by the system. The value of the external name is assigned using the 'VALUE OF FILE NAME IS' clause in File Section of Data Division. Here the name can be a literal or a variable name. If the name used is a variable an assignment to it before an 'OPEN' clause results in opening the file. This property is utilised in user files. File name GENEL is defined as general user file in the program, the file name taken from the query is assigned to it as the external file name.

Another property is sending control instructions from the program to the operating system MCP (19), using the 'CALL SYSTEM ZIP' command. This facility is used to change the name of the workfile 'ÇIKTIKÜT' to 'EÇIKTI', programatically.

4.2. The Structure Of The Software

The DRS Software System consists of a main program called the monitor, four basic procedures for each of the retrieval operations and several other commonly used procedures.

All of the commonly used procedures are defined as COBOL sections providing segmentation. Segmentation is a facility permitting the user to physically subdivide the Procedure Division of a COBOL object program (14). The segments are independent of each other and can be overlayed by other independent segments. The high-order digit of the 4-digit section number specifies the overlay area in to which that segment may be placed. Each overlay area is the size of the largest segment assigned to it. When a segment that is not in memory is invoked, that segment overlays the segment

previously occupying its overley area. This facility of COBOL, decreases the amount of memory required by the execution of the program by overlaying the procedures that are not used at the same time.

Input to the DRS software is from the translator part (11) of the total DDS software system, which is not yet completed. Therefore the input is assumed to be converted to tokens (See Appendix A). These tokens are read from punched cards, and are assumed to have no errors.

In figure 4.1 the general structure of the software system is given. The main program part "ÇEK" invokes one of the retrieval procedures; or the auxiliary procedures depending on the value part of the first token.

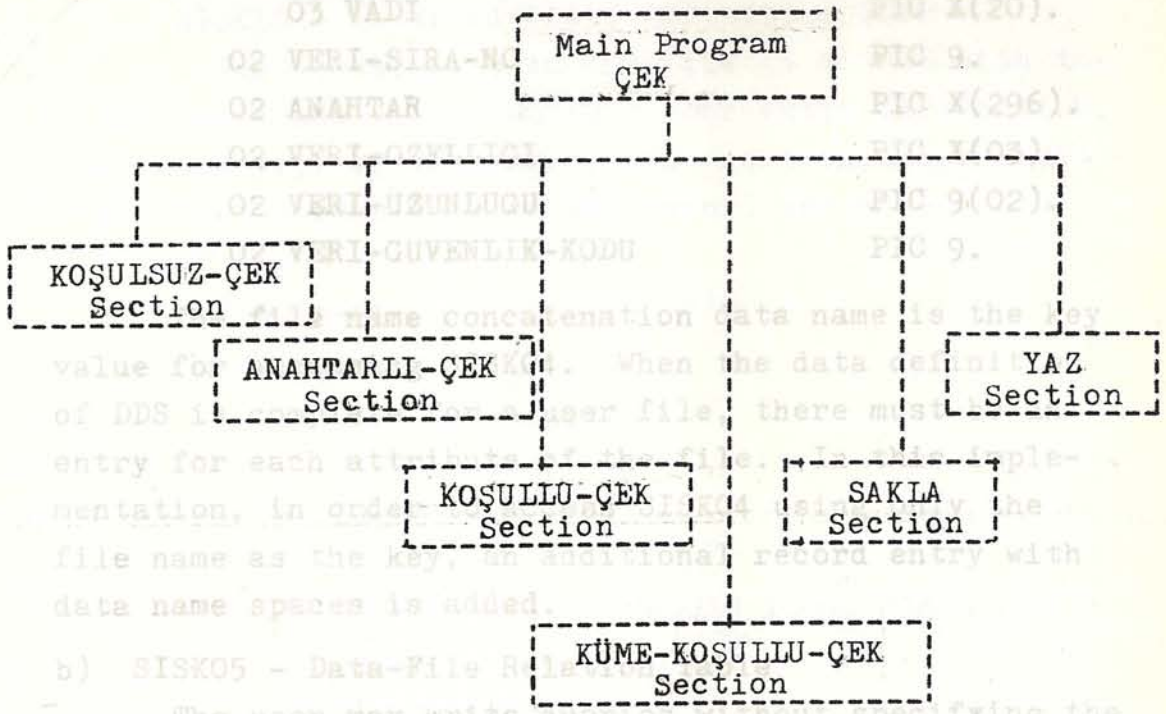

Figure 4.1 General structure of the data retrieval sublanguage software system.

4.2.1. Data Structures

The basic data structures of the system is the indexed organized system files, introduced in Section 2.2. The other data structures are the temporary storage areas between the user program and the system files.

4.2.1.1. The System Files

The designed DDS software system has a total of fourteen system files (7), but only six of them are used in the retrieval part.

a) SISKO4 - Record Structure Definition File

The specifications of each data attribute kept in the files are stored in this system file.

The record format of SISKO4 is :

```
01 TUTANAK4.
   02 KADI-VADI.
      03 KADI                          PIC X(15).
      03 VADI                          PIC X(20).
   02 VERI-SIRA-NO                     PIC 9.
   02 ANAHTAR                          PIC X(296).
   02 VERI-OZELLIGI                    PIC X(03).
   02 VERI-UZUNLUGU                    PIC 9(02).
   02 VERI-GUVENLIK-KODU               PIC 9.
```

The file name concatenation data name is the key value for accessing SISKO4. When the data definition of DDS is complete for a user file, there must be an entry for each attribute of the file. In this implementation, in order to access SISKO4 using only the file name as the key, an additional record entry with data name spaces is added.

b) SISKO5 - Data-File Relation Table

The user may write queries without specifying the file name from which the data is to be retrieved.

Finding the name of the file in which the data item is stored, is the responsibility of the DDS system. SISK05 is used for this purpose; for each data item in the database there is an entry in the file.

The record format of SISK05 is :

```
01 TUTANAK5.
   02 VERI-ADI-5                    PIC X(20).
   02 KUTUK-ADLARI                  PIC X(159).
```

c) SISK07 - Hash Function Guide

The hash function name for each file is stored in SISK07. In the implementation, a prime number which is the bucket size is added to the attributes; this will be used in key conversion algorithms.

The record format of SISK07 is :

```
01 TUTANAK7.
   02 KUTUK-ADI-7                   PIC X(15).
   02 D-F-MODUL-ADI                 PIC X(15).
   02 BOLEN                         PIC 9(03).
```

d) SISK10 - Inverted Lists Definition File

The access path to the file is stored with the data name and value couple. Some extra information, such as type of inversion, and creation,usage count and last access date to the record, are also stored in this file.

The record format of SISK10 is :

```
01 TUTANAK10.
   02 VERI-ADI-DEGERI.
      03 VERI-ADI                   PIC X(15).
      03 VERI-DEGERI                PIC X(66).
   02 EYOLU.
      03 ERISIM-YOLU OCCURS 10000 TIMES PIC 9.
   02 DEVIRME-TURU                  PIC X(05).
   02 OLUSMA-SEKLI                  PIC X(07).
   02 KULLANIM-SAYISI               PIC 9(04).
   02 SON-KULLANIM-TARIHI           PIC 9(06).
```

e) SISK11 - Key Lists File

The lists of keys for each file in the system are stored in SISK11. The key values are used to directly access the user files.

The record format of SISK11 is :

```
01 TUTANAK11.
   02 KUTUK-ADI-11                PIC X(15).
   02 ANAHTAR-LISTESI             PIC X(10000).
```

f) SISK12 - Sort File

SISK12 provides the sorting of a file by one or a maximum of four keys without changing the physical order of the records. The relative positions of the records which are stored in sorted order are mapped to the keys, permitting access to the user file in sorted order.

The record format of SISK12 is :

```
01 TUTANAK12.
   02 KADI-SANAH-SDUZEN.
      03 KUTUK-ADI-12              PIC X(15).
      03 SIRALAMA-ANAHTARI-1       PIC X(20).
      03 SIRALAMA-ANAHTARI-2       PIC X(20).
      03 SIRALAMA-ANAHTARI-3       PIC X(20).
   02 SIRALAMA-DUZENI              PIC X(06).
   02 SIRA-ZINCIRI                 PIC X(10000).
   02 S-SON-KULLANIM-TARIHI        PIC 9(06).
   02 S-KULLANIM-SAYISI            PIC 9(04).
```

4.2.1.2. Other Files Used in the Implementation

a) GENEL

Access to different user files with different record sizes and structures will be possible by using the general user file called GENEL. The external file name is the name of the user file. This name is assigned to variable GENELKADI when the need arises to open the file.

```
01 GENTUT                         PIC X(100).
```

b) CIKTIKUT

The design of Uçkan (7) creates a workarea called ÇIKTIKUT as a result of a retrieval statement. All of the retrieved data items are stored in this file.

```
01 CIKTITUT.
   02 CIKTIT OCCURS 132 TIMES        PIC X.
```

c) ECIKTI

As a result of a retrieval statement the file CIKTIKUT is created and this file is copied to ECIKTI. This used in retrieval statements which operate on the previous workfile.

```
01 ECIKTUT.
   02 ECIK OCCURS 132 TIMES          PIC X.
```

d) KART

Tokens which are input to the data retrieval programs are taken from this file. Each record contains twenty tokens composed of a type and a value part.

```
01 SIMGELER.
   02 SIMGE OCCURS 20 TIMES.
      03 TUR                        PIC 9(02).
      03 DEGER                      PIC 9(02).
```

e) CIKTI

CIKTI is a printer file used for printing the contents of the workarea.

```
01 CIKIS                            PIC X(132).
```

4.2.1.3. Temporary Data Structures

a) VOZ. This is the one dimensional numeric array in which the position, length, and key specifications of all of the data attributes in the user file are stored. SISK04 is read and the data attributes data-sequence-no, type and length are used to calculate the beginning position of each one in the file.

```
01 VOZ.
   02 VO OCCURS 99 TIMES.
      03 VERIAD                    PIC X(20).
      03 VERIPOZ                   PIC 9(03).
      03 VERIUZUN                  PIC 9(03).
      03 VERIANAH                  PIC 9.
```

b) VERI-OZEL. Using the array VOZ, the specifications of data attributes that will be retrieved is stored in VERI-OZEL. The record format is the same as VOZ.

```
01 VERI-OZEL.
   02 VOZELLIGI OCCURS 99 TIMES.
      03 VERIADI                   PIC X(20).
      03 VERIBAS                   PIC 9(03).
      03 VERIUZ                    PIC 9(03).
      03 VERIAN                    PIC 9.
```

c) CIKTI-VOZEL. The specifications of data which is placed to CIKTIKUT is stored in the array. The purpose of keeping this information is for retrieving some data from the previously created file, and also when the save statement 'SAKLA' is used. The record format is the same as VOZ.

```
01 CIKTI-VOZEL.
   02 CVOZELLIGI OCCURS 99 TIMES.
      03 CVERIADI                  PIC X(20).
      03 CVERIBAS                  PIC 9(03).
      03 CVERIUZ                   PIC 9(03).
      03 CVERISAKLA                PIC 9.
```

d) ACIKAN. This is character array used to store the key values that will be used to access the user file. The keys are taken from SISK11.

```
01 ACIKAN
   02 ACIKANAH OCCURS 999 TIMES    PIC X(20).
```

e) KOSULLAR. This file is used in the processing of conditional retrieval statements to store the data attribute, relational operator, and the data values.

```
01 KOSULLAR.
   02 KOSUL-VERI OCCURS   2 TIMES   PIC X(20).
   02 ISLEC      OCCURS   2 TIMES   PIC 9.
   02 KOSUL-ANAH OCCURS   2 TIMES   PIC X(66).
```

f) SON-EYOLU, ENSON-EYOLU. These two data structures are used in determining the access path using the conditions and the values in SISKlO

```
01 SON-EYOLU.
   02 SON-ERISIM-YOLU OCCURS 250 TIMES PIC 9.
01 ENSON-EYOLU.
   02 ENSON-ERISIM-YOLU OCCURS 250 TIMES PIC 9.
```

g) TIRE-ANAH. This is a one dimensional array for storing the range of key values used in Keyed Retrieval statements. The initial and final values are stored in BASLA, and BITIR. If there exists a single key value then BITIR will be empty.

```
01 TIRE-ANAH.
   02 BASLA OCCURS   20 TIMES      PIC X(20).
   02 BITIR OCCURS   20 TIMES      PIC X(20).
```

4.2.2. Functions of System Component

In this section the functions of the basic procedures will be given. When a retrieval program begins with 'BAŞLA', the control is given to the main routine 'CEK' as a monitor. The monitor then gives control to the retrieval procedures. The program goes to the end-of-job when the finishing phrase 'BİTİR' is reached.

a) CEK. This is the main procedure of the software system. Depending on the tokens, one of the retrieval procedures defined below, or one of the auxiliary procedures can be invoked by CEK. When control returns back, the workarea is copied to another file and the next statement of the retrieval program is executed.

b) KOSULSUZ-CEK. This is the procedure for unconditional retrieval. It deals with queries for retrieving all or some of the data attributes in a user file. The user may not specify the name of the file that the retrieval will be done from; in this case the name is determined by the software system. In this procedure, it is optional to select the data items in sorted order.

c) ANAHTARLI-CEK. This is the procedure for keyed retrieval. It is used for retrieving all or some of the data attributes when the open key values are given. These key values may specify a range. May be single values or a set of keys that were retrieved in a previous retrieval statement.

d) KOSULLU-CEK. This is the procedure for conditional retrieval. As input it gets the data name and the data value with the appropriate relational operator token between; the negation of the condition can also be achieved. As a result of this procedure the data attributes that satisfy the given conditions will be retrieved.

e) KUME-KOSULLU-CEK. This is the procedure for set conditional retrieval. It is used to select the attributes satisfying the conditions of the set in the workarea. These may be conditions for equality, inclusion, and exclusion.

f) SAKLA. This procedure is invoked when the auxiliary statement for saving some of the data values placed in the workarea, by a previous retrieval statement is to be processed. It marks the data values which will be saved.

g) YAZ. This procedure prints the contents of the workarea at the end of a DDS retrieval program.

h) Procedures for Obtaining Names

The main function of these procedures is to obtain the names of elements that are used in queries. These procedures are listed below with their functions.

(i) KUTUK-ADI-AL : for obtaining the file name

(ii) VERI-ADI-AL : for obtaining the data names

(iii) KOSUL-ANAH-AL : for obtaining the data name of a condition

(iv) KOSUL-VERI-AL : for obtaining the data value of a condition

(v) ISLECI-AL : for obtaining the relational operator of a condition

(vi) KUME-ADI-AL : for obtaining the set name in the workarea

The class of the token, listed in Appendix B, shows the type of the dataname and the value is the pointer to list containing the datanames. This is the assumed output of the translator (11).

i) Procedures shared by the others

In addition to the ones given above, there are procedures which are used for various purposes. These form a group of shared procedures; the names and functions of these procedures are given below.

(i) VERI-OZELLIGI-AL. The specifications such as name, beginning position, and length of data that will be retrieved is determined and an array is created with these values. The keys and the length of the keys of the user file are also determined.

(ii) ANAHTARLARI-AL. The key values stored in system file SISK11 are read and placed into an array. These will be used in accesses to the file. In order to get the records in sorted order, the key values are arranged according to the sort sequence information given in the system file SISK12.

(iii) ANAHTAR-KONTROL. If the data attribute to be retrieved is a key value which is stored in an array, then the access to the user file is avoided. The values in the array are taken. The function of this procedure is providing this.

(iv) KUTUKTEN-CEK. This procedure is for retrieving the records from the user file. It transforms the key values to relative record numbers by using a specified hash function.

(v) KUTUK-ADI-BUL. If the file name from which the data attributes are to be retrieved is not specified in the query, this procedure is used to find the name.

(vi) ERISIM-YOLU-AL. This procedure is used to determine the access path to access the records satisfying the conditions in a conditional retrieval statement.

(vii) ANAHTAR-BIRDEN-FAZLA. If there is more than one key, then the keys are taken from the system file SISK11, by this procedure.

(viii) ESIT-ANAHTARLARI-AL. The keys which are satisfying the set equality condition with the key values in the workarea are retrieved.

(ix) DISINDAKI-ANAHTARLARI-AL. The keys which do not satisfy the set equality condition with the key values in the workarea are retrieved.

(x) ICEREN-ANAHTARLARI-AL. The keys which include the key values in the workarea are retrieved.

(xi) TUTANAK-OKU. This procedure is used to read the record whose relative record number is given. The required data attribute is retrieved and placed into the workarea.

(xii) ANAH-DONUSTUR. The conversion of the alphanumeric key to a numeric string, ready for a hashing algorithm is the purpose of this procedure.

(xiii) ANAHTAR-PARCALA. If the user file has more than one key, this procedure separates the keys.

j) Procedures for Implementing Hash Functions

The hash functions tranform the symbolic keys to relative positions in the user files (5, 6). A function which is satisfactory for one particular user file may not be as good for another. In this implementation the user is given the possibility of choosing one of three hashing algorithms using the methods of division, shifting and folding. The standad names DF1, DF2, and DF3 (7) are used in the system file SISKO7, to represent respectively the three procedures :

(i) BOL for implementing the division algorithm,
(ii) KAYDIR for implementing the shifting algorithm,
(iii) KATLA for implementing the folding algorithm.

4.3. Algorithms

In this section the algorithms for the four basic retrieval procedures are given. The types and values of tokens used in the algorithms are specified in Appendix A.

a) Algorithm for Unconditional Retrieval
('Koşulsuz ÇEK')

S1 . If TYPE = 7 get the file name (perform KUTUK-ADI-AL).
S2 . If TYPE = 8 get the data names to be retrieved (perform VERI-AL).
S3 . If the file name was not specified find it (perform KUTUK-ADI-BUL).
S4 . If TYPE = 14 get the sort order of records on the specified keys (perform SIRA-ANAHTARI-AL).
S5 . Perform VERI-OZELLIGI-AL and get the specifications of the data that will be retrieved.
S6. Perform ANAHTARLARI-AL, get all of the keys to access to the user file.

S7 . Are the data names to be retrieved keys ?
If so get them from the array in ANAHTAR-KONTROL,
if not perform KUTUKTEN-CEK.

S8 . If TYPE = 20 it is the end of this procedure.

b) Algorithm for Keyed Retrieval
('Anantarlı ÇEK').

S1 . If TYPE = 7 get the file name (KUTUK-ADI-AL).

S2 . If TYPE = 9 get the open key list by performing
ACIK-ANAHTAR-AL.

S3 . If TYPE = 10 , the key values in the workarea
will be used, so these are stored in ACIKAN by
performing KUME-ANAHTAR.

S4 . If TYPE = 8 get data names performing VERI-AL,
else go (S2).

S5 . If the file name is not specified perform KUTUK-
ADI-BUL.

S6 . Get the specifications of the data to be retrieved
(perform VERI-OZELLIGI-AL).

S7 . If ACIKAN is empty (set name not specified), get
all of the key values (perform ANAHTARLARI-AL),
else if there is more than one key perform
ANAHTAR-BIRDEN-FAZLA and then ESIT-ANAHTAR-AL.

S8 . Perform ANAHTAR-KONTROL.

S9 . If the data to be retrieved are not only keys,
then perform KUTUKTEN-CEK.

S10. If TYPE = 20 end of the procedure.

c) Algorithm for conditional retrieval
('Koşullu Çek')

S1 . If TYPE = 7 get the file name .

S2 . If TYPE = 8 get the data attributes of the condi-
tion, KOSUL-VERI-AL.

S3 . If TYPE = 11 get the relational operator, ISLECI-
AL.

S4 . If TYPE = 9 get the data value performing KOSUL-ANAH-AL.

S5 . If TYPE = 13; if VALUE = 3 get the relational operator (VE, or VEYA) go S3, else it is negation (NOT) perform OLUMSUZ-YAP.

S6 . If TYPE = 8 get the data names to be retrieved, perform VERI-AL.

S7 . If the file name isnot specified perform KUTUK-ADI-BUL.

S8 . Perform VERI-OZELLIGI-AL, get the specifications of the data.

S9 . Perform ANAHTARLARI-AL, get the key values.

S10. Get the access path satisfying the conditions (perform ERISIM-YOLU-AL).

S11. Now map the access path to the key values, having only the required keys in hand (perform EYOLU-ANAH-AL).

S12. Perform ANAHTAR-KONTROL.

S13. If the data to be retrieved are not keys perform KUTUKTEN-CEK.

S14. If TYPE = 20 it is the end of the procedure.

Algorithm for Set Conditional Retrieval
('Küme Koşullu ÇEK')

S1 . If TYPE = 7 perform KUTUK-ADI-AL.

S2 . If TYPE = 8 perform KOSUL-VERI-AL, get the name of data that will satisfy the condition in the file.

S3 . If TYPE = 12 get the condition value (EŞİT, İÇERİR or DIŞINDAKİ).

S4 . If TYPE = 10 get the set name and the values by performing KUME-ANAHTAR.

S5 . If TYPE = 8 get the data names (perform VERI-AL).

S6 . If the file name isnot specified perform KUTUK-ADI-BUL.

S7 . Perform VERI-OZELLIGI-AL.

S8 . Perform ANAHTARLARI-AL.

S9 . If CIFTLI-KUME-KOSULU = 1 get the keys satisfying the set equality (ESIT-ANAHTARLARI-AL).

S10. If CIFTLI-KUME-KOSULU = 2 get the keys satisfying the set inclusion (ICEREN-ANAHTARLARI-AL).

S11. If CIFTLI-KUME-KOSULU = 0 get the keys satisfying the set exclusion (DISINDAKI-ANAHTARLARI-AL).

S12. Using the key values get the data to be retrieved.

S13. If TYPE = 20 it is the end of the procedure.

5. SUMMARY AND DISCUSSION

5.1. Summary

The aim of this study is to implement a software system for the data retrieval facility of a Dynamic Database Management System (DDS) defined by Uçkan (7).

In the thesis a summary of the data languages of some specific database management systems based on the three models, namely the relational, hierarchical and network models, are given. The DDS is reviewed in terms of its basic features, architecture and elements, data language and data retrieval facility available. Later a syntactic notation is introduced, and lexical elements are identified. Using these definitions, syntactic and semantic issues of DDS data retrieval sublanguage are given. This sublanguage consists of the variations of the four basic retrieval statements qualified as being unconditional, keyed, conditional, and set conditional.

The software developed in COBOL for the sublanguage is introduced. Specificly COBOL data retrieval facilities are analyzed, the structure of the software pertaining to data structures and functions of system components are expained. Algorithms developed during implementation are also expained and application examples are supplied.

5.2. Discussion

The DDS data retrieval program may consist of retrieval statements written in a specific order (See Section 3.3). There are four types of retrieval statements ; unconditional, keyed, conditional and set conditional.

In the implementation, in addition to these four retrieval statements, namely save and print, were fully implemented. During this implementation several assumptions, extensions and limitations on the original design were incorporated :

(i) In conditional queries it is assumed that the access paths have already been created in file SISK10. The explicit inversion ('DEVİR') of the user file is the responsibility of the DMS (Data Manipulation Sublanguage) which is yet to be implemented (12).

(ii) The unconditional statement with the sort option assumes that the records are already sorted and created in SISK12. If not, the explicit sorting routine ('SIRALA') of DMS should be invoked by the DRS software. However, since DMS is not yet implemented, this has not been incorporated into DRS.

(iii) There are three hashing methods defined in the system to provide access to the user files. If a different hashing method is to be used, its definition must be added to the sofware, and the necassary changes in the system file SISK07 and the corresponding software module 'DAGITIM-FONKSIYONU-AL'.

(iv) The access paths for conditional queries are kept as digit strings, although the original design (7) calls for bit strings. This revision was necessary because of the limitations of the language of implementation.

(v) It is possible to access the files SISK04 and SISK10 using either a single key (primary key) or two keys concatenated (primary and secondary keys). In order to provide for this change in the design of the two files were made and an extra record containing only the file name was added to SISK04 and an extra record containing only the data name was added to SISK10.

The work carried out in this thesis can be extended in several ways :

(i) The auxiliary statements used for gathering statistics, namely count and find, can be implemented, once the ambiguity in their design is resolved.

(ii) Other hashing methods can be incorporated in order to provide a wide range of selection for the users.

(iii) When the translator of DDS is completed (11), the effected procedures of DRS can be updated to incorporate any change.

(iv) When the DMS is completed (12), the routines of DRS which is needed to invoke, the inversion and sorting operations can be rewritten without any assumptions.

LIST OF REFERENCES

1. CARDENAS,A.F., Data Base Management Systems, Allyn and Bacon,Inc.,Boston,Massachusetts,U.S.A., 1979.

2. DATE,C.J. , An Introduction to Database Systems, 2 nd Edition,Addison-Wesley Publishing Company,Inc.,Reading,Massachusetts,U.S.A., 1977.

3. MARTIN,J. , Computer Data-Base Organization, 2 nd Edition,Prentice-Hall,Inc.,Englewood Cliffs, N.J.,U.S.A, 1977.

4. TSICHRITZIS,D.C., LOCHOVSKY,F.H., Data Base Management Systems, Academic Press,Inc.,N.Y.,U.S.A, 1977.

5. KNUTH,D.E., The Art of Computer Programming, Volume 3, Sorting and Searching, Addison-Wesley Publishing Company,Inc.,Reading,Massachusetts, U.S.A., 1973.

6. HOROWITZ,E., SAHNI,S., Fundamentals of Data Structures, Pitman Publishing Limited,London,England, 1978.

7. UÇKAN,Y. , Design of a Relational Data Language and Data Base Management System, (in Turkish), Habilitation Thesis,METU,Ankara, 1980.

8. ÖZKARAHAN,E.A., et al, Database Management System Facilities at METU, Report no IS-DB.8, Dept. of Comp. Eng.,METU,Ankara, 1979.

9. ASTRAHAN,M.M., et al, "System R: A Relational Approach to Data Base Management", ACM Transactions on Database Systems, Vol 1, No.2, 1976.

10. UZDİL,P., Design and Implementation of a Software System for the Data Definition Sublanguage for a Dynamic DBMS, Master Thesis, Dept. of Comp. Eng.,METU,Ankara, 1980.

11. KESKİN,M., Design and Implementation of a Translator for DDS- A Dynamic DBMS, Master Thesis, Dept. of Comp. Eng.,METU,Ankara, (in preparation).

12. BÜYÜKBAYRAM,A., Design and Partial Implementation of a Data Manipulation Sublanguage for DDS- A Dynamic DBMS, Master Thesis, Dept. of Comp. Eng.,METU,Ankara, (in preparation).

13. TROUT,F., MEYER,D., B 2000/B 3000/B 4000 DMS II Implementation Considerations, TIP 1114238-016, Burroughs Corp.,Detroit,Michigan, U.S.A., 1981.

14. _____, B 2000/B 3000/B 4000 Series COBOL ANSI-74 Reference Manual, No.1090735, Burroughs Corp.,Detroit,Michigan,U.S.A., 1980.

15. _____, B 2000/B 3000/B 4000 Series DMS II User's Manual, No.1108925, Burroughs Corp.,Detroit, Michigan,U.S.A., 1980.

16. _____, B 7000/B 6000 Series DMS II DASDL Reference Manual, No.5001480, Burroughs Corp.,Detroit, Michigan,U.S.A., 1978.

17. _____, B 7000/B 6000 Series DMS II INQUIRY Reference Manual, No.5001472, Burroughs Corp.,Detroit, Michigan,U.S.A., 1977.

18. _____, IBM System/360 Operating System PL/I (F) Language Reference Manual, IBM Corp.,New York, N.Y.,U.S.A., 1972.

19. _____, B 2000/B 3000/B 4000 Series MCPVI System Software Operation Guide, Vol. 1, No.1127529 Burroughs Corp.,Detroit,Michigan,U.S.A., 1981.

APPENDICES

APPENDIX A. LEXICAL TOKENS OF DRS

The tokens expected by the DRS sofware consists of two parts : type and value. Value is the pointer to the value list which is assumed to be created on disk by the translator part of the implementation (11). Below the numbers that represent the type and value in the program is listed.

TYPE :	VALUE :
03 - Retrieval statement	01 - Unconditional retrieval
	02 - Keyed retrieval
	03 - Conditional retrieval
	04 - Set conditional retrieval
04 - Save statement	00
05 - Write statement	00
07 - File name	PN - Pointer to name list
08 - Data name	PN - Pointer to name list
09 - Open key	PV - Pointer to value list
10 - Set name	PN - Pointer to name list
11 - Relational operator	01 - =
	02 - <
	03 - >
	04 - <=
	05 - >=
	06 - ≠
	07 - —
12 - Set condition couple	00 - DIŞINDAKİ
	01 - EŞİT
	02 - İÇERİR
13 - Boolean operator	01 - ∧
	02 - ∨
	03 - ¬
14 - Sort key	PN - Pointer to name list
20 - End-of-statement	00

APPENDIX B
SAMPLE RUNS

```
/*  BASLA :  */
/*  ÖĞRENCİ KÜTÜĞÜNDEN TÜM VERİLERİ ÇEK  */
030107060800200C
/*  YAZ.
06002000
```

ÖĞRENCİ-NO	SOYADI-ADI	FAKÜLTE	BÖLÜM	YIL
3650	ÖDEMİŞLİ ALİ	ENG	CS	2
3665	CAN NAZAN	ENG	CS	1
3804	UYGAR CEM	FAS	MAN	2
3805	ASLI ÇMER	ENG	EE	2
3860	KALAÇ ALİ	ARCH	CP	1
4003	ÖZTUNALI AYŞE	ENG	CS	2
4010	KIRAN İSMAİL	ENG	ME	1
4035	TARIK BİLGE	ARCH	CP	1
4200	SARI FÜSUN	ENG	EE	2
4225	AKLI HÜSEYİN	ENG	CS	1
4316	ŞİMŞEK FÜRUZAN	ARCH	ARCH	1

```
/*  BİTİR.  */
```

```
/*  BAŞLA :  */
/*  DERS KÜTÜĞÜNDEN TÜM VERİLERİ SEK.  */
0301070108002000
/*  YAZ.  */
06002000
```

DERS-KODU	DERS-ADI	KREDİ	TİP	ÖĞR-ÜYE-ADI
CS112	INTR.TO COMPUTERS	09	BÖLÜM	ÖNCÜ
CS122	MATHEMATICAL LOGIC	09	BÖLÜM	İLERİ
CS214	DATA STRUCTURES	09	BÖLÜM	ÖNCÜ
CS222	PROBABILITY AND STAT.	09	BÖLÜM	BORLU
CS252	DATA PROCESSING	12	GENEL	CEMAL
ENG104	EXPOSITORY WRITING	12	GENEL	ADİL
MATH152	CALCULUS	12	GENEL	CİVAN
PHYS106	GENERAL PHYSICS	15	GENEL	KARA

```
/*  BİTİR.  */
```

```
/*  BAŞLA :  */
/*  DEĞERLENDİRME KÜTÜĞÜNDEN TÜM VERİLERİ ÇEK  */
03010703080020 00
/*  YAZ.
06002000
```

ÖĞRENCİ-NO	DERS-KODU	NOT
3650	CS214	AA
3650	CS222	BA
3650	CS252	AA
3665	CS112	BB
3665	CS122	CB
3804	CS252	FF
3804	ENG104	FD
3804	MATH152	CC
3805	CS214	AA
4003	CS214	CB
4003	CS222	CC
4003	ENG104	DC
4010	CS252	W
4035	CS252	AA
4035	ENG104	W
4035	PHYS106	BA
4200	CS252	CC
4200	ENG104	CC
4200	MATH152	CB
4200	PHYS106	FF
4225	CS112	W
4225	CS122	FF
4225	ENG104	FD
4225	MATH152	DD
4225	PHYS106	CB
4316	ENG104	AA
4316	MATH152	AA
4316	PHYS106	AA

```
/*  BİTİR.  */
```

```
/*  BAŞLA :  */
/*  DERS-KODU + DERS-ADI VERILERINI ÇEK.  */
02010802080820OC
/*  YAZ.  */
06002000

DERS-KODU   DERS-ADI
CS112       INTR.TO COMPUTERS
CS122       MATHEMATICAL LOGIC
CS214       DATA STRUCTURES
CS222       PROBABILITY AND STAT.
CS252       DATA PROCESSING
ENG104      EXPOSITORY WRITING
MATH152     CALCULUS
PHYS106     GENERAL PHYSICS
/*  BITIR.  */
```

```
/*  BAŞLA :  */
/*  ÖĞRENCİ KÜTÜĞÜNDEN TÜM VERİLERİ                    */*/
/*  FAKÜLTE,BÖLÜM ANAHTARINA GÖRE SİRALI ÇEK.          */*/
03010706080014101411200008050042000
/*  YAZ.                                                */
06002000
```

ÖĞRENCİ-NO	SOYADI-ADI	FAKÜLTE	BÖLÜM	YIL
4316	ŞİMŞEK FÜRUZAN	ARCH	ARCH	1
3860	KALAÇ ALİ	ARCH	CP	1
4035	TARIK BİLGE	ARCH	CP	1
3650	ÖDEMİŞLİ ALİ	ENG	CS	2
3665	CAN NAZAN	ENG	CS	1
4003	ÖZTUNALI AYŞE	ENG	CS	2
4225	AKLI HÜSEYİN	ENG	CS	1
3805	ASLI ÖMER	ENG	EE	2
4200	SARI FÜSUN	ENG	EE	2
4010	KIRAN İSMAİL	ENG	ME	1
3804	UYGAR CEM	FAS	MAN	2

```
/*  BİTİR.  */
```

```
/*  BAŞLA :  */
/*  ÖĞRENCİ KÜTÜĞÜNDEN ( BÖLÜM = CS ) OLMAYAN */
/*  SOYADI-ADI,ÖĞRENCİ-NO VERİLERİNİ ÇEK.      */
030307060811110109011303080508042000
/*  YAZ.
06002000
```

SOYADI-ADI	ÖĞRENCİ-NO
UYGAR CEM	3804
ASLI ÖMER	3805
KALAŞ ALİ	3860
KIRAN İSMAİL	4010
TARIK BİLGE	4035
SARI FÜSUN	4200
ŞİMŞEK FÜRUZAN	4316

```
/*  BİTİR.  */
```

```
/*  BAŞLA :  */
/*  DEĞERLENDİRME KÜTÜĞÜNDEN DERS-KODU = CS214 KOŞULLU  */
/*  ÖĞRENCİ-NO VERİSİNİ ÇEK.                            */
030307030802110109060042000
/*  ÖĞRENCİ KÜTÜĞÜNDEN ( ÖĞRENCİ-NO ) ANAHTARLI         */
/*  SOYADI-ADI VERİSİNİ ÇEK.                            */
030207061004080520000
/*  YAZ.  */
06002000

DERS-ADI
SOYADI-ADI
ÖDEMİŞLİ ALİ
ASLI ÖMER
ÖZTUNALI AYŞE
/*  BİTİR.  */
```

```
/*  BASLA :  */
/*  DEGERLENDIRME KUTUGUNDEN OGRENCI-NO = 3650 KOSULLU  */
/*  DERS-KODU VERISINI SEK.                             */
030307030804110109100802200O
/*  ( DERS-KODU ) ANAHTARLI DERS-ADI VERISINI SEK.      */
0302100208082000
/*  YAZ.
06002000

DERS-ADI

DATA STRUCTURES

PROBABILITY AND STAT.

DATA PROCESSING
/*  BITIR.  */
```

```
/*  BAŞLA :  */
/*  DEĞERLENDİRME KÜTÜĞÜNDEN DERS-KODU = CS214 KOŞULLU */
/*  ÖĞRENCİ-NO,DERS-KODU VERİLERİNİ ÇEK.                */
030307030802110109060804080220000
/*  ÖĞRENCİ-NO VERİSİNİ SAKLA.                          */
040008042000
/*  ( DERS-KODU ) ANAHTARLI DERS-ADI VERİSİNİ ÇEK.      */
030210020808200 0
/*  YAZ.
06002000

ÖĞRENCİ-NO  DERS-ADI

3650        DATA STRUCTURES

3805        DATA STRUCTURES

4003        DATA STRUCTURES
/*  BİTİR.  */
```

```
/*  BAŞLA :  */
/*  DEĞERLENDIRME KÜTÜĞÜNDEN ÖĞRENCI-NO = 3650 KOŞULLU  */
/*  DERS-KODU VERISINI   ŞEK.                           */
030307030804110109100802200O
/*  DEĞERLENDIRME KÜTÜĞÜNDEN ( DERS-KODU ) DIŞINDAKI    */
/*  ÖĞRENCI-NO VERISINI ŞEK.                            */
0304070308021200100208042000
/*  YAZ.  */
06002000


ÖĞRENCI-NO

3665

4225

4316
/*  BITIR.  */
```

APPENDIX C

PROGRAM LISTING OF THE SOFTWARE DEVELOPPED

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ANAPRG.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. B-2900.
OBJECT-COMPUTER. B-2900.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CIKTI     ASSIGN TO PRINTER.
    SELECT CIKTIKUT  ASSIGN TO DISK.
    SELECT DEGERLER  ASSIGN TO DISK.
    SELECT VDEGERL   ASSIGN TO DISK.
    SELECT ECIKTI    ASSIGN TO DISK.
    SELECT GENEL     ASSIGN TO DISKPACK
       ACCESS RANDOM
       ACTUAL KEY IS GENANAH STATUS DFS.
    SELECT KART      ASSIGN TO READER.
    SELECT SISK04    ASSIGN TO DISKPACK
       ORGANIZATION INDEXED ACCESS DYNAMIC
       RECORD KEY KADI-VADI STATUS DFS.
    SELECT SISK05    ASSIGN TO DISKPACK
       ORGANIZATION INDEXED ACCESS RANDOM
       RECORD KEY VERI-ADI-5 STATUS DFS.
    SELECT SISK07    ASSIGN TO DISKPACK
       ORGANIZATION INDEXED ACCESS RANDOM
       RECORD KEY KUTUK-ADI-7 STATUS DFS.
    SELECT SISK10    ASSIGN TO DISKPACK
       ORGANIZATION INDEXED ACCESS DYNAMIC
       RECORD KEY VERI-ADI-DEGERI STATUS DFS.
    SELECT SISK11    ASSIGN TO DISKPACK
       ORGANIZATION INDEXED ACCESS RANDOM
       RECORD KEY KUTUK-ADI-11 STATUS DFS.
```

```
     SELECT SISK12   ASSIGN TO DISKPACK
          ORGANIZATION INDEXED ACCESS RANDOM
            RECORD KEY KADI-SANAH-SDUZEN
            ALTERNATE RECORD KEY SIRALAMA-DUZENI WITH DUPLICATES
            FILE STATUS DFS.
 DATA DIVISION.
 FILE SECTION.
 FD   CIKTI.
 01   CIKIS                                    PIC X(132).
 FD   CIKTIKUT.
 01   CIKTITUT.
      02 CIKTIT              OCCURS 132 TIMES PIC X.
*****************************************************************
*     THE FOLLOWING TWO FILES ARE ASSUMED                       *
*     TO BE CREATED BY THE DDS TRANSLATOR                       *
*                                                               *
 FD   DEGERLER.
 01   DLIST.
      02 DEGERLIST           OCCURS  20 TIMES PIC X(20).
 FD   VDEGERL.
 01   VDLIST.
      02 VDEGERLIST          OCCURS  20 TIMES PIC X(20).
*****************************************************************
 FD   ECIKTI.
 01   ECIKTUT.
      02 ECIK                OCCURS 132 TIMES PIC X.
 FD   GENEL VALUE OF FILENAME  IS GENELKADI.
 01   GENTUT                                   PIC X(100).
 FD   KART.
 01   SIMGELER.
      02 SIMGE               OCCURS  20 TIMES.
         03 TUR                                PIC 9(02).
```

```
           03 DEGER                                  PIC 9(02).
      *    ***************************************************************
      *    *                                                             *
      *    *                  S Y S T E M   F I L E S                    *
      *    *                                                             *
      *    ***************************************************************
      *    *                        S I S K 0 4                          *
      *    ***************************************************************
       FD  SISK04 VALUE OF FILENAME IS "KB04SK",
           AREASIZE IS 25, AREAS IS 1.
       01  TUTANAK4.
           02 KADI-VADI.
              03 KADI                               PIC X(15).
              03 VADI                               PIC X(20).
           02 VERI-SIRA-NO                          PIC 9.
           02 ANAHTAR                               PIC X(296).
           02 VERI-OZELLIGI                         PIC X(03).
           02 VERI-UZUNLUGU                         PIC 9(02) .
           02 VERI-GUVENLIK-KODU                    PIC 9.
      *    ***************************************************************
      *    *                        S I S K 0 5                          *
      *    ***************************************************************
       FD  SISK05 VALUE OF FILENAME  IS     "KB05SK",
           AREASIZE IS 20, AREAS IS 1.
       01  TUTANAK5.
           02 VERI-ADI-5                            PIC X(20).
           02 KUTUK-ADLARI                          PIC X(159).
      *    ***************************************************************
      *    *                        S I S K 0 7                          *
      *    ***************************************************************
       FD  SISK07 VALUE OF FILENAME  IS     "KB07SK",
           AREASIZE IS 10, AREAS IS 1.
```

```
 01  TUTANAK7.
     02 KUTUK-ADI-7                          PIC X(15).
     02 D-F-MODUL-ADI                        PIC X(15).
     02 BOLEN                                PIC 9(03).
*    ***************************************************************
*    *                      S I S K 1 0                              *
*    ***************************************************************
 FD  SISK10 VALUE OF FILENAME  IS     "KB10SK",
         AREASIZE IS 40, AREAS IS 1.
 01  TUTANAK10.
     02 VERI-ADI-DEGERI.
        03 VERI-ADI                          PIC X(15).
        03 VERI-DEGERI                       PIC X(66).
     02 EYOLU.
        03 ERISIM-YOLU  OCCURS 10000 TIMES PIC 9.
     02 DEVIRME-TUR                          PIC X(05).
     02 OLUSMA-SEKLI                         PIC X(07).
     02 KULLANIM-SAYISI                      PIC 9(04) .
     02 SON-KULLANIM-TARIHI                  PIC 9(06).
*    ***************************************************************
*    *                      S I S K 1 1                              *
*    ***************************************************************
 FD  SISK11 VALUE OF FILENAME  IS     "KB11SK",
         AREASIZE IS 10, AREAS IS 1.
 01  TUTANAK11.
     02 KUTUK-ADI-11                         PIC X(15).
     02 ANAHTAR-LISTESI                      PIC X(10000).
*    ***************************************************************
*    *                      S I S K 1 2                              *
*    ***************************************************************
 FD  SISK12 VALUE OF FILENAME  IS     "KB12SK",
         AREASIZE IS 20, AREAS IS 1.
```

```
01  TUTANAK12.
    02 KADI-SANAH-SDUZEN.
       03 KUTUK-ADI-12                          PIC X(15).
       03 SIRALAMA-ANAHTARI-1                   PIC X(20).
       03 SIRALAMA-ANAHTARI-2                   PIC X(20).
       03 SIRALAMA-ANAHTARI-3                   PIC X(20).
       03 SIRALAMA-ANAHTARI-4                   PIC X(20).
    02 SIRALAMA-DUZENI                          PIC X(06).
    02 SIRA-ZINCIRI                             PIC X(10000).
    02 S-SON-KULLANIM-TARIHI                    PIC 9(06).
    02 S-KULLANIM-SAYISI                        PIC 9(04).
WORKING-STORAGE SECTION.
77  ANAHSAY                                     PIC 9(04) COMP.
77  ANAHSAYC                                    PIC 9(04) COMP.
77  ANAH-UZUNLUGU                               PIC 9(03) COMP.
77  BOSVERI                    VALUE   SPACES   PIC X(20).
77  CIFTLI-KUME-KOSULU                          PIC 9(06) COMP.
77  C2                                          PIC X(66).
77  C37                                         PIC X(39) VALUE
    "0123456789ABCÇDEFGĞHIİJKLMNOÖPRSŞTUÜVYZ".
77  DEGERI                                      PIC X(20).
77  DISINDAKI                                   PIC 9     COMP.
77  DL                                          PIC 9(02) COMP.
77  GENANAH                                     PIC 9(08) COMP.
77  GENELKADI                                   PIC X(06).
77  HATA-KODU                                   PIC 9(03) COMP.
77  I                                           PIC 9(06) COMP.
77  ISNG                                        PIC 9     COMP.
77  J                                           PIC 9(02) COMP.
77  K                                           PIC S99   COMP.
77  KA                                          PIC X(15).
77  KEY1                                        PIC X.
```

```
77  KL                              PIC S99 .COMP.
77  KUMEADI                        PIC X(20).
77  KUME-ANAHUZ                    PIC 9(02).
77  KUTUKADI                       PIC X(15).
77  KUTUKADIYOK                    PIC 9 .COMP.
77  KX                              PIC X(15).
77  L                               PIC 9(02) COMP.
77  LL                              PIC 9(08) COMP.
77  LK                              PIC 9(08) COMP.
77  M                               PIC 9(02) COMP.
77  MANTIKSAL-ISLEC                 PIC 9 .COMP.
77  N                               PIC 9(03) COMP.
77  NL                              PIC 9(02) COMP.
77  NN                              PIC 9(02) COMP.
77  NX                              PIC 9(06) COMP.
77  OLUMSUZ                         PIC 9 .COMP.
77  P                               PIC 9(06) COMP.
77  R                               PIC 9(06) COMP.
77  RAKAM                           PIC 9(02).
77  REM                             PIC 9(08) COMP.
77  S                               PIC 9(03) COMP.
77  SAKLAYIN                        PIC 9 .COMP.
77  SAYAC                           PIC 9(15) COMP.
77  SIRAANAH                        PIC 9(02) COMP.
77  SIRABASI                        PIC 9(02) COMP.
77  SIRASONU                        PIC 9(02) COMP.
77  SIRALA                          PIC 9 .COMP.
77  SIRALI                          PIC 9 .COMP.
77  SONANAH                         PIC 9 .COMP.
77  SONUC                           PIC 9(08) COMP.
77  SWT1                            PIC 9.
77  TARIH                           PIC 9(06) COMP.
```

```
77  TEKANH                 USAGE IS DISPLAY PIC X(20).
77  TUM                                    PIC 9(66) COMP.
77  UA                                      PIC 9(08) COMP.
77  VDEGERI                                PIC X(66).
77  VERISAY                                PIC 9(02) COMP.
77  VERISAYC                               PIC 9(02) COMP.
77  VERISAYE                               PIC 9(02) COMP.
01  ACIKAN.
    02 ACIKANAH           OCCURS 999 TIMES PIC X(20).
01  ACIKANC.
    02 ACIKANAHC          OCCURS  99 TIMES PIC X(20).
01  ANAHTS.
    02 ANAHT              OCCURS   4 TIMES PIC X(06).
01  ANAH-VERI.
    02 AV                 OCCURS 296 TIMES PIC X.
01  ANAH-UZUN.
    02 ANAHUZ             OCCURS  10 TIMES PIC 99.
01  C1.
    02 C1X                OCCURS  66 TIMES PIC X.
01  CIKTI-VOZEL.
    02 CVOZELLIGI         OCCURS  99 TIMES.
       03 CVERIADI                         PIC X(20).
       03 CVERIBAS                         PIC 9(03).
       03 CVERIUZ                          PIC 9(03).
       03 CVERISAKLA                       PIC 9.
01  DFS.
    02 DFS1                                PIC X.
    02 DFS2                                PIC X.
01  ENSON-EYOLU.
    02 ENSON-ERISIM-YOLU OCCURS 250 TIMES PIC 9.
01  KOSULLAR.
    02 KOSUL-VERI         OCCURS   2 TIMES PIC X(20).
```

```
    02 ISLEC                OCCURS   2 TIMES PIC 9.
    02 KOSUL-ANAH           OCCURS   2 TIMES PIC X(66).
01  NU.
    02 NUM                  OCCURS  20 TIMES PIC 99.
01  NUMMS.
    02 NUMM                 OCCURS  20 TIMES PIC 9.
01 RAKAM-LAR.
    02 RAKAMLAR             OCCURS  20 TIMES PIC 9.
01  SIRASI.
    02 SIRA                 OCCURS 100 TIMES PIC 99.
01  SIRAVERI.
    02 SIRA-VERI            OCCURS   4 TIMES PIC X(20).
01  SON-EYOLU.
    02 SON-ERISIM-YOLU      OCCURS 250 TIMES PIC 9.
01  TIRE-ANAH.
    02 BASLA                OCCURS  20 TIMES PIC X(20).
    02 BITIR                OCCURS  20 TIMES PIC X(20).
01  TIRELER.
    02 TIRE                 OCCURS  20 TIMES PIC X.
01  VOZ.
    02 VO                   OCCURS  99 TIMES.
       03 VERIAD                              PIC X(20).
       03 VERIPOZ                             PIC 9(03).
       03 VERIUZUN                            PIC 9(03).
       03 VERIANAH                            PIC 9.
01  VERI-OZEL.
    02 VOZELLIGI            OCCURS  99 TIMES.
       03 VERIADI                             PIC X(20).
       03 VERIBAS                             PIC 9(03).
       03 VERIUZ                              PIC 9(03).
       03 VERIAN                              PIC 9.
*
```

```
*
*     ****************************************************************
*     *                                                              *
*     *                    M A I N   P R O G R A M                   *
*     *                          C E K                               *
*     *                                                              *
*     ****************************************************************
 PROCEDURE DIVISION.
 CEK SECTION.
 CS.
     OPEN INPUT KART,OUTPUT CIKTI.
     OPEN INPUT DEGERLER,VDEGERL.
     MOVE 0 TO NL,SAKLAYIN.
     OPEN I-O SISK04,SISK05,SISK07,SISK10,SISK11,SISK12.
 CSX.
     PERFORM ILK-DEGER-VER.
     PERFORM KARTI-OKU.
     IF TUR (P) NOT = 3
         PERFORM CEKME
         GO CSX.
     IF DEGER (P) = 1 PERFORM KOSULSUZ-CEK
     ELSE IF DEGER (P) = 2 PERFORM ANAHTARLI-CEK
     ELSE IF DEGER (P) = 3 PERFORM KOSULLU-CEK
     ELSE IF DEGER (P) = 4 PERFORM KUME-KOSULLU-CEK.
     CALL SYSTEM ZIP USING "PFM LST1 CIKTIK DSK A SAVE.".
     WAIT 60.
     CALL SYSTEM ZIP USING  "RM ECIKTI.".
     CALL SYSTEM ZIP USING "CH CIKTIK TO ECIKTI.".
     GO CSX.
 CEKME.
     IF TUR (P) = 4 PERFORM SAKLA
     ELSE IF TUR (P) = 6 PERFORM YAZ.
```

```
DEGER-AL.
    MOVE DEGER (P) TO LL.
    MOVE DEGERLIST (LL) TO DEGERI.
VDEGER-AL.
    MOVE DEGER (P) TO LL.
    MOVE VDEGERLIST (LL) TO VDEGERI.
ISLECI-AL.
    MOVE DEGER (P) TO ISLEC (ISNO).
    PERFORM SIMGE-AL.
KARTI-OKU.
    READ KART AT END GO BITIRIN.
    IF TUR (1) = "/*" OR TUR (1) = "  "
        WRITE CIKIS FROM SIMGELER
        GO KARTI-OKU.
    MOVE 0 TO P.
    PERFORM SIMGE-AL.
    WRITE CIKIS FROM SIMGELER.
KOSUL-ANAH-AL.
    PERFORM VDEGER-AL.
    MOVE VDEGERI TO KOSUL-ANAH (ISNO).
    PERFORM SIMGE-AL.
KOSUL-VERI-AL.
    ADD 1 TO ISNO.
    PERFORM DEGER-AL.
    MOVE DEGERI TO KOSUL-VERI (ISNO).
    PERFORM SIMGE-AL.
HATA.
    DISPLAY HATA-KODU ,"/HATAVAR".
    GO BITIRIN.
KOD-KONTROL.
    IF DFS1 = 2 AND DFS2 = 2 NEXT SENTENCE
    ELSE DISPLAY " HATA KODU : ",DFS GO BITIRIN.
```

```
    DISPLAY "KOD-KONTROL".
SISK-HATA.
    DISPLAY "SISK",HATA-KODU," HATALI OKUMA".
    GO BITIRIN.
SIMGE-AL.
    ADD 1 TO P.
ILK-DEGER-VER.
    MOVE 0 TO SIRALA,SIRALI,ISNO,ANAHSAY,P.
    MOVE SPACES TO ACIKAN,KOSULLAR,SIRAVERI,TIRE-ANAH,TIRELER.
    MOVE 1 TO KUTUKADIYOK.
    MOVE 0 TO MANTIKSAL-ISLEC,ENSON-EYOLU,KUME-ANAHUZ.
    READ DEGERLER AT END GO BITIRIN.
    READ VDEGERL  AT END GO BITIRIN.
KUTUK-ADI-AL.
    IF P > 20 PERFORM KARTI-OKU.
    PERFORM DEGER-AL.
    MOVE DEGERI TO KUTUKADI.
    MOVE 0 TO KUTUKADIYOK.
    PERFORM SIMGE-AL.
VERI-AL.
    MOVE ZEROES TO VERI-OZEL.
    MOVE 0 TO VERISAY,TUM.
    IF DEGER (P) = 0
     IF KUTUKADIYOK = 1
      MOVE 2 TO HATA-KODU
      PERFORM HATA
     ELSE MOVE 1 TO TUM
      PERFORM SIMGE-AL
    ELSE PERFORM VERI-ADI-AL UNTIL TUR (P) NOT = 8.
VERI-ADI-AL.
    IF P > 20 PERFORM KARTI-OKU.
    ADD 1 TO VERISAY.
```

```
    PERFORM DEGER-AL.
    MOVE DEGERI TO VERIADI (VERISAY).
    PERFORM SIMGE-AL.
CIKTIPOZBUL.
    SUBTRACT 1 FROM K GIVING L.
    MOVE VERIADI (I) TO CVERIADI (K).
    MOVE VERIUZ  (I) TO CVERIUZ  (K).
    IF K = 1 MOVE 1 TO CVERIBAS (K)
    ELSE ADD CVERIBAS (L),CVERIUZ (L) GIVING CVERIBAS (K).
    ADD 1 TO K.
VERIYI-BUL.
    IF CVERIADI (I) = KUMEADI MOVE I TO RAKAM
       ADD 1 , VERISAYC GIVING I.
KUME-ANAHTAR.
    PERFORM KUME-ADI-AL.
    PERFORM VERIYI-BUL VARYING I FROM 1 BY 1 UNTIL I > VERISAYC.
    MOVE VERISAYC TO VERISAYE.
    MOVE CVERIBAS (RAKAM) TO N.
    ADD CVERIUZ (RAKAM),N GIVING LL.
    MOVE CVERIUZ (RAKAM) TO KUME-ANAHUZ.
    OPEN INPUT ECIKTI.
    READ ECIKTI AT END MOVE 1 TO SWT1.
    MOVE 0 TO ANAHSAY,SWT1.
    PERFORM ECIKTI-OKU UNTIL SWT1 = 1.
    CLOSE ECIKTI DISMISS.
KUME-ADI-AL.
    PERFORM DEGER-AL.
    MOVE DEGERI TO KUMEADI.
    PERFORM SIMGE-AL.
ECIKTI-OKU.
    PERFORM ECIK-ANAHTAR-AL.
    READ ECIKTI AT END MOVE 1 TO SWT1.
```

```
ECIK-ANAHTAR-AL.
    ADD 1 TO ANAHSAY.
    MOVE 1 TO LK.
    PERFORM ECIK-AKTAR VARYING I FROM N BY 1 UNTIL I = LL.
    MOVE TIRELER TO ACIKANAH (ANAHSAY).
    MOVE SPACES TO TIRELER.
ECIK-AKTAR.
    MOVE ECIK (I) TO TIRE (LK).
    ADD 1 TO LK.
BITIRIN.
    CLOSE SISK04 LOCK,
          SISK05 LOCK,
          SISK07 LOCK,
          SISK10 LOCK,
          SISK11 LOCK,
          SISK12 LOCK.
    CLOSE DEGERLER,
          VDEGERL.
    STOP RUN.
CEK-SON. EXIT.
*    ****************************************************************
*    *                                                              *
*    *                UNCONDITIONAL RETRIEVAL                       *
*    *                K O S U L S U Z - C E K                       *
*    *                                                              *
*    ****************************************************************
 KOSULSUZ-CEK SECTION 1051.
 KSUZ-CEK.
    PERFORM SIMGE-AL.
    IF TUR (P) = 0 PERFORM KARTI-OKU.
    IF TUR (P) = 7 PERFORM KUTUK-ADI-AL.
    IF TUR (P) = 8 PERFORM VERI-AL
```

```
    ELSE MOVE 5 TO HATA-KODU PERFORM HATA.
    IF KUTUKADIYOK = 1
       PERFORM KUTUK-ADI-BUL.
    IF TUR (P) = 14 PERFORM SIRA-ANAHTARI-AL.
    IF SIRALA NOT = 1
    PERFORM VERI-OZELLIGI-AL.
    MOVE SPACES TO TIRE-ANAH.
    PERFORM ANAHTARLARI-AL.
    PERFORM ANAHTAR-KONTROL.
    IF SWT1 NOT = 1
       PERFORM KUTUKTEN-CEK.
    IF TUR (P) NOT = 20 MOVE 3 TO HATA-KODU
       PERFORM HATA.
    GO KSUZCEK-CIK.
SIRA-ANAHTARI-AL.
    MOVE 0 TO SIRAANAH.
    MOVE 1 TO KL.
    PERFORM ANAHTAR-AL UNTIL TUR (P) NOT = 14.
    STRING KUTUKADI,SIRA-VERI (1),SIRA-VERI (2),SIRA-VERI (3),
      SIRA-VERI (4),"ARTAN " DELIMITED BY SIZE
      INTO KADI-SANAH-SDUZEN.
    READ SISK12 KEY KADI-SANAH-SDUZEN INVALID KEY GO SIRALAMA.
    MOVE 1 TO SIRALI.
    MOVE SIRA-ZINCIRI TO SIRASI.
    ACCEPT TARIH FROM DATE.
    MOVE TARIH TO S-SON-KULLANIM-TARIHI.
    ADD 1 TO S-KULLANIM-SAYISI.
    REWRITE TUTANAK12 INVALID KEY MOVE 12 TO HATA-KODU
       PERFORM SISK-HATA.
ANAHTAR-AL.
    IF P > 20 PERFORM KARTI-OKU.
    ADD 1 TO SIRAANAH.
```

```
     PERFORM DEGER-AL.
     MOVE DEGERI TO SIRA-VERI (SIRAANAH)
     PERFORM SIMGE-AL.
 SIRALAMA.
     MOVE 1 TO SIRALA.
     PERFORM VERI-OZELLIGI-AL.
     ADD 1,VERISAY GIVING SIRABASI.
     PERFORM ANAHTARLARI-AL.
     PERFORM KUTUKTEN-CEK.
 KSUZCEK-CIK. EXIT.
*    ****************************************************************
*    *                                                              *
*    *                        KEYED RETRIEVAL                       *
*    *                A N A H T A R L I - C E K                     *
*    *                                                              *
*    ****************************************************************
 ANAHTARLI-CEK SECTION 1052.
 ANAH-CEK.
     PERFORM SIMGE-AL.
     IF TUR (P) = 7 PERFORM KUTUK-ADI-AL.
     PERFORM ADONGU UNTIL TUR (P) = 8.
     IF TUR (P) = 8 PERFORM VERI-AL
     ELSE MOVE 5 TO HATA-KODU PERFORM HATA.
     IF KUTUKADIYOK = 1
        PERFORM KUTUK-ADI-BUL.
     PERFORM VERI-OZELLIGI-AL.
     IF ACIKANAH (1) = SPACES
        PERFORM ANAHTARLARI-AL
     ELSE
        IF KUME-ANAHUZ < ANAH-UZUNLUGU
           PERFORM ANAHTAR-BIRDEN-FAZLA
           ADD 1 , VERIUZ (1) GIVING LK.
```

```
           PERFORM ESIT-ANAHTAR-AL UNTIL SONANAH = 1.
      PERFORM ANAHTAR-KONTROL.
      IF SWT1 NOT = 1
         PERFORM KUTUKTEN-CEK.
      IF TUR (P) NOT = 20 MOVE 3 TO HATA-KODU
         PERFORM HATA.
      GO ANAH-CEK-CIK.
ADONGU.
      IF TUR (P) = 9 PERFORM ACIK-ANAHTAR-AL
         UNTIL TUR (P) NOT = 9.
      IF TUR (P) = 10 PERFORM KUME-ANAHTAR.
ACIK-ANAHTAR-AL.
      ADD 1 TO ANAHSAY.
      PERFORM VDEGER-AL.
      MOVE DEGERI TO BASLA (ANAHSAY), BITIR (ANAHSAY).
      PERFORM SIMGE-AL.
      IF TUR (P) = 11 AND DEGER (P) = 7
         PERFORM SIMGE-AL
         IF TUR (P) NOT = 9
            MOVE 6 TO HATA-KODU
            PERFORM HATA
         ELSE PERFORM VDEGER-AL
            MOVE DEGERI TO BITIR (ANAHSAY).
ESIT-ANAHTAR-AL.
      PERFORM ANAHTAR-PARCALA.
      PERFORM ESITI-AKTAR VARYING I FROM 1 BY 1
         UNTIL I > ANAHSAYC.
ESITI-AKTAR.
      IF ACIKANAHC (I) = DEGERI
         ADD 1 TO ANAHSAY
         ADD ANAHSAYC TO I
         MOVE TEKANH TO ACIKANAH (ANAHSAY).
```

```
 ANAH-CEK-CIK. EXIT.
*    ****************************************************************
*    *                                                              *
*    *                                                              *
*    *                  CONDITIONAL RETRIEVAL                       *
*    *                   K O S U L L U - C E K                      *
*    *                                                              *
*    ****************************************************************
 KOSULLU-CEK SECTION 1053.
 KLU-CEK.
     PERFORM SIMGE-AL.
     IF TUR (P) =  0 PERFORM KARTI-OKU.
     IF TUR (P) =  7 PERFORM KUTUK-ADI-AL.
 KOSUL-AL.
     IF TUR (P) =  8 PERFORM KOSUL-VERI-AL.
     IF TUR (P) = 11 PERFORM ISLECI-AL.
     IF TUR (P) =  9 PERFORM KOSUL-ANAH-AL.
     IF TUR (P) = 13
        IF DEGER (P) NOT = 3
          MOVE DEGER (P) TO MANTIKSAL-ISLEC
          PERFORM SIMGE-AL
          GO KOSUL-AL
        ELSE MOVE 1 TO OLUMSUZ
           PERFORM SIMGE-AL.
     IF TUR (P) = 8 PERFORM VERI-AL
     ELSE MOVE 5 TO HATA-KODU PERFORM HATA.
     IF KUTUKADIYOK = 1
        PERFORM KUTUK-ADI-BUL.
     PERFORM VERI-OZELLIGI-AL.
     PERFORM ANAHTARLARI-AL.
     PERFORM ERISIM-YOLU-AL
        VARYING I FROM 1 BY 1 UNTIL I > ISNO.
     IF OLUMSUZ = 1
```

```
           PERFORM OLUMSUZ-YAP VARYING K FROM 1 BY 1
             UNTIL K > ANAHSAY.
       MOVE 0 TO LL.
       PERFORM EYOLU-ANAH-AL VARYING K FROM 1 BY 1 UNTIL K > ANAHSAY.
       MOVE LL TO ANAHSAY.
       PERFORM ANAHTAR-KONTROL.
       IF SWT1 NOT = 1
          PERFORM KUTUKTEN-CEK.
       IF TUR (P) NOT = 20 MOVE 3 TO HATA-KODU
          PERFORM HATA.
       GO TO KLU-CIK.
 OLUMSUZ-YAP.
       IF ENSON-ERISIM-YOLU (K) = 1
       MOVE 0 TO ENSON-ERISIM-YOLU (K)
       ELSE MOVE 1 TO ENSON-ERISIM-YOLU (K).
 EYOLU-ANAH-AL.
       IF ENSON-ERISIM-YOLU (K) = 1
          ADD 1 TO LL
          MOVE ACIKANAH (K) TO ACIKANAH (LL).
 KLU-CIK. EXIT.
*      ***************************************************************
*      *                                                             *
*      *                  SET-CONDITIONAL RETRIEVAL                  *
*      *          K U M E - K O S U L L U - C E K                    *
*      *                                                             *
*      ***************************************************************
 KUME-KOSULLU-CEK SECTION 1054.
 KKCEK.
       PERFORM SIMGE-AL.
       IF TUR (P) =  0 PERFORM KARTI-OKU.
       IF TUR (P) = 7 PERFORM KUTUK-ADI-AL.
       IF TUR (P) = 8 PERFORM KOSUL-VERI-AL
```

```
           MOVE 0 TO DISINDAKI
           IF TUR (P) = 12 MOVE DEGER (P) TO CIFTLI-KUME-KOSULU
           PERFORM SIMGE-AL.
       IF TUR (P) = 10 PERFORM KUME-ANAHTAR.
       IF TUR (P) =  8 PERFORM VERI-AL
       ELSE MOVE 5 TO HATA-KODU PERFORM HATA.
       IF KUTUKADIYOK = 1
          PERFORM KUTUK-ADI-BUL.
       PERFORM VERI-OZELLIGI-AL.
       PERFORM ANAHTAR-BIRDEN-FAZLA.
       MOVE 0 TO NL,SWT1.
       PERFORM KOSULU-ARA VARYING I FROM 1 BY 1
          UNTIL I > VERISAY.
       IF CIFTLI-KUME-KOSULU = 1
          PERFORM ESIT-ANAHTARLARI-AL UNTIL SONANAH = 1
       ELSE
          IF CIFTLI-KUME-KOSULU = 2
             PERFORM ICEREN-ANAHTARLARI-AL UNTIL SONANAH = 1
          ELSE PERFORM DISINDAKI-ANAHTARLARI-AL UNTIL SONANAH = 1.
       PERFORM ANAHTAR-KONTROL.
       IF SWT1 NOT = 1
          PERFORM KUTUKTEN-CEK.
       IF TUR (P) NOT = 20 MOVE 3 TO HATA-KODU
          PERFORM HATA.
       GO TO KKCIK.
   ESIT-MI.
       IF ACIKANAHC (I) = DEGERI
          ADD 1 TO NL
          ADD ANAHSAYC TO I.
   KOSULU-ARA.
       IF VERIAD (I) = KOSUL-VERI (1)
          MOVE VERIPOZ (I) TO J
```

```
           MOVE VERIUZUN (I) TO LK
           ADD VERISAY TO I.
       KKCIK. EXIT.
      *    ****************************************************************
      *    *                                                              *
      *    *          THIS PROCEDURE SAVES THE DATA ATTRIBUTES STORED     *
      *    *          INTO THE WORKAREA BY A PREVIOUS RETRIEVAL STATEMENT *
      *    *                                                              *
      *    ****************************************************************
       SAKLA SECTION 1055.
       SAKLA-BAS.
           MOVE 1 TO SAKLAYIN.
           PERFORM SIMGE-AL.
           IF TUR (P) = 8 PERFORM VERI-AL
           ELSE MOVE 5 TO HATA-KODU PERFORM HATA.
           PERFORM VERIYI-BUL VARYING K FROM 1 BY 1 UNTIL K > VERISAY
               AFTER I FROM 1 BY 1 UNTIL I > VERISAYC.
           MOVE VERISAYC TO VERISAYE.
           GO SAKLA-CIK.
       VERIYI-BUL.
           IF VERIADI (K) = CVERIADI (I)
               MOVE 1 TO CVERISAKLA (I)
               ADD VERISAYC TO I
           ELSE MOVE 0 TO CVERISAKLA (I).
       SAKLA-CIK.
           EXIT.
      *    ****************************************************************
      *    *                                                              *
      *    *          THIS PROCEDURE PRINTS THE DATA VALUES               *
      *    *        STORED IN THE WORKAREA BY A RETRIEVAL STATEMENT       *
      *    *                                                              *
      *    ****************************************************************
```

```
YAZ SECTION 1057.
YAZ-BAS.
    MOVE 1 TO S.
    MOVE SPACES TO CIKIS.
    PERFORM BASLIK-YAZ VARYING I FROM 1 BY 1 UNTIL I > VERISAYE.
    WRITE CIKIS AFTER ADVANCING 3.
    MOVE SPACES TO CIKIS.
    MOVE 0 TO L.
    OPEN INPUT ECIKTI.
    MOVE 0 TO SWT1.
    PERFORM CIKTI-OKU THRU CIKTI-YAZMA UNTIL SWT1 = 1.
    CLOSE ECIKTI DISMISS.
    GO YAZ-SON.
BASLIK-YAZ.
    MOVE CVERIADI (I) TO DEGERI.
    MOVE 0 TO L.
    INSPECT DEGERI TALLYING L FOR CHARACTERS
      BEFORE INITIAL " ".
    IF L < CVERIUZ (I) MOVE CVERIUZ (I) TO VERIUZ (I)
    ELSE MOVE L TO VERIUZ (I).
    MOVE CVERIADI (I) TO DEGERI.
    MOVE S TO N.
    STRING DEGERI FOR L INTO CIKIS
      WITH POINTER N.
    ADD VERIUZ (I) TO S.
    ADD 2 TO S.
CIKTI-OKU.
    READ ECIKTI AT END MOVE 1 TO SWT1
      GO CIKTI-YAZMA.
    MOVE 1 TO N,S.
    PERFORM SATIR-YAZ VARYING I FROM 1 BY 1
      UNTIL I > VERISAYE.
```

```
     WRITE CIKIS AFTER ADVANCING 2.
     MOVE SPACES TO CIKIS.
 CIKTI-YAZMA. EXIT.
 SATIR-YAZ.
     MOVE SPACES TO DEGERI.
     MOVE CVERIBAS (I) TO K.
     UNSTRING ECIKTUT INTO DEGERI FOR CVERIUZ (I)
        WITH POINTER K.
     MOVE S TO L.
     STRING DEGERI FOR VERIUZ (I) INTO CIKIS
        WITH POINTER L.
     ADD VERIUZ (I) TO S. ADD 2 TO S.
 YAZ-SON. EXIT.
*    ****************************************************************
*    *                                                              *
*    *          THIS PROCEDURE OBTAINS THE CHARACTERISTICS          *
*    *           OF THE DATA ATTRIBUTES TO BE RETRIEVED             *
*    *                                                              *
*    ****************************************************************
 VERI-OZELLIGI-AL SECTION 2051.
 VERIOZELLIGIAL.
     MOVE ZEROES TO VOZ.
     STRING KUTUKADI BOSVERI DELIMITED BY SIZE INTO KADI-VADI.
     READ SISK04 KEY KADI-VADI INVALID KEY MOVE 4 TO HATA-KODU
                                   PERFORM SISK-HATA.
     MOVE 0 TO K.
     READ SISK04 NEXT AT END PERFORM KOD-KONTROL.
     MOVE ANAHTAR TO ANAH-VERI.
     PERFORM TARA UNTIL VADI = BOSVERI.
     MOVE 1 TO VERIPOZ (1).
     PERFORM VERIPOZBUL VARYING J FROM 2 BY 1 UNTIL J > K.
     MOVE 1 TO LL.
```

```
    MOVE 0 TO ANAH-UZUNLUGU.
    PERFORM ANAHTAR-TARA UNTIL AV (LL) = SPACES.
    MOVE 0 TO ANAHUZ (LL),KL.
    IF SIRALI = 1 ADD SIRAANAH TO KL.
    IF TUM NOT = 1
       ADD VERISAY TO KL
       MOVE VERISAY TO VERISAYC
       PERFORM DEGERBUL VARYING J FROM 1 BY 1 UNTIL J > KL
    ELSE PERFORM DEGERAKTR VARYING J FROM 1 BY 1 UNTIL J > K
       MOVE K TO VERISAYC.
    MOVE K TO VERISAY.
    GO TO VERIOZELLIGIAL-CIK.
TARA.
    ADD 1 TO K.
    IF TUM NOT = 1
       PERFORM ARA VARYING I FROM 1 BY 1 UNTIL I > VERISAY.
    MOVE VADI TO VERIADI (VERI-SIRA-NO).
    MOVE VERI-UZUNLUGU TO VERIUZUN (VERI-SIRA-NO).
    READ SISK04 NEXT AT END MOVE SPACES TO VADI.
ARA.
    IF VADI = VERIADI (I)
       MOVE VERI-SIRA-NO TO VERIBAS (I).
    IF VADI = KOSUL-VERI (1) MOVE VERI-SIRA-NO TO N.
VERIPOZBUL.
    SUBTRACT 1 FROM J GIVING L.
    ADD VERIUZUN (L),VERIPOZ (L) GIVING VERIPOZ (J).
ANAHTAR-TARA.
    MOVE AV (LL) TO RAKAM.
    MOVE 1 TO VERIANAH (RAKAM).
    MOVE VERIUZUN (RAKAM) TO ANAHUZ (LL).
    ADD ANAHUZ (LL) TO ANAH-UZUNLUGU.
    ADD 1 TO LL.
```

```
DEGERBUL.
    IF SIRALA = 1 MOVE SIRABASI TO KL
       PERFORM SARA VARYING I FROM 1 BY 1 UNTIL I > SIRAANAH
       SUBTRACT 1 FROM KL GIVING SIRASONJ.
    MOVE VERIBAS (J) TO L.
    MOVE VERIPOZ (L) TO VERIBAS (J).
    MOVE VERIUZUN (L) TO VERIUZ (J).
    MOVE VERIANAH(L) TO VERIAN(J).
SARA.
    IF VADI = SIRA-VERI (I)
       MOVE VADI TO VERIADI (LL)
       MOVE VERI-SIRA-NO TO VERIBAS (KL)
       ADD 1 TO KL.
DEGERAKTR.
    MOVE VERIAD   (J) TO VERIADI (J).
    MOVE VERIPOZ  (J) TO VERIBAS (J).
    MOVE VERIUZUN (J) TO VERIUZ  (J).
    MOVE VERIANAH (J) TO VERIAN  (J).
VERIOZELLIGIAL-CIK. EXIT.
*    ***************************************************************
*    *                                                             *
*    *          THIS PROCEDURE OBTAINS THE KEY VALUES              *
*    *          OF THE USER FILE THAT WILL BE ACCESSED             *
*    *                                                             *
*    ***************************************************************
ANAHTARLARI-AL SECTION 2052.
ANAHTARLARIAL.
    MOVE KUTUKADI TO KUTUK-ADI-11.
    READ SISK11 KEY KUTUK-ADI-11 INVALID KEY MOVE 11 TO HATA-KODU
                                 PERFORM SISK-HATA.
    MOVE 0 TO ANAHSAY,SONANAH.
    MOVE 1 TO NX.
```

```
    MOVE SPACES TO ACIKAN.
    IF SIRALI = 1
      PERFORM SIRA-ZINCIRI-AL VARYING KL FROM 1 BY 1
        UNTIL SIRA (KL) = 0
    ELSE
      IF BASLA (1) NOT = SPACES
        MOVE 1 TO LK
        PERFORM ACIK-ANAHTARLARI-AL UNTIL BASLA (LK) = SPACES
      ELSE
        PERFORM ANAHTAR-AL-DONUSTUR THRU ANAHTAR-AL-CIK
          UNTIL SONANAH = 1.
    GO TO ANAHTARLARIAL-CIK.
SIRA-ZINCIRI-AL.
    SUBTRACT 1 FROM SIRA (KL) GIVING NX.
    MULTIPLY ANAH-UZUNLUGU BY NX.
    ADD 1 TO NX.
    PERFORM ANAHTAR-AL-DONUSTUR THRU ANAHTAR-AL-CIK.
ACIK-ANAHTARLARI-AL.
    MOVE SPACES TO TEKANH.
    PERFORM ANAHTAR-AL-DONUSTUR UNTIL
      TEKANH NOT < BASLA (LK) OR SONANAH = 1.
    PERFORM ANAHTARI-AKTAR.
    IF BASLA (LK) NOT = BITIR (LK)
      PERFORM ANAHTAR-AL-DONUSTUR THRU ANAHTAR-AL-CIK
        UNTIL TEKANH NOT < BITIR (LK) OR SONANAH = 1.
    IF TEKANH > BITIR (LK) MOVE SPACES TO ACIKANAH (ANAHSAY)
      SUBTRACT 1 FROM ANAHSAY.
    ADD 1 TO LK.
    IF TEKANH > BASLA (LK) MOVE 1 TO NX.
ANAHTAR-AL-DONUSTUR.
    MOVE SPACES TO TEKANH.
    MOVE NX TO LL.
```

```
    UNSTRING ANAHTAR-LISTESI  INTO TEKANH FOR ANAH-UZUNLUGU
      WITH POINTER LL.
    IF TEKANH = SPACES MOVE 1 TO SONANAH GO ANAHTAR-AL-CIK.
    ADD ANAH-UZUNLUGU TO NX.
ANAHTARI-AKTAR.
    ADD 1 TO ANAHSAY.
    MOVE TEKANH TO ACIKANAH (ANAHSAY).
ANAHTAR-AL-CIK. EXIT.
ANAHTARLARIAL-CIK. EXIT.
*    ***************************************************************
*    *                                                             *
*    *     THIS PROCEDURE CONTROLS IF THE DATA RETRIEVED           *
*    *     IS KEYS, IF SO THEY WILL BE TAKEN FROM KEY.LIST         *
*    *                                                             *
*    ***************************************************************
ANAHTAR-KONTROL SECTION 2053.
ANAHTARKONTROL.
    MOVE 1 TO DL,K.
    MOVE 0 TO SWT1.
    PERFORM ANAH-KONTROL UNTIL VERIAN (DL) NOT = 1.
    IF SAKLAYIN =1 OPEN INPUT ECIKTI.
    IF DL > VERISAYC
      MOVE 1 TO SWT1
      MOVE SPACES TO C2
      OPEN OUTPUT CIKTIKUT
      MOVE SPACES TO CIKTITUT
      PERFORM ANAHLISTTEN-AL VARYING LK FROM 1 BY 1
        UNTIL LK > ANAHSAY
      CLOSE CIKTIKUT DISMISS
      MOVE 1 TO K
      PERFORM CIKTIPOZBUL VARYING I FROM 1 BY 1
        UNTIL I > VERISAYC.
```

```
    IF SAKLAYIN = 1 CLOSE ECIKTI DISMISS.
    GO TO ANAHTARKONTROL-CIK.
ANAH-KONTROL.
    ADD 1 TO DL.
ANAHLISTTEN-AL.
    MOVE 1 TO R.
    IF SAKLAYIN = 1 PERFORM ECIKTIDAN-AKTAR.
    PERFORM ANAHLIST-PARCALA VARYING S FROM 1 BY 1
       UNTIL S > VERISAYC.
    WRITE CIKTITUT.
ANAHLIST-PARCALA.
    MOVE SPACES TO C1.
    MOVE VERIBAS (S) TO LL.
    MOVE ACIKANAH (LK) TO C2.
    UNSTRING C2 INTO C1 FOR VERIUZ (S) WITH POINTER LL.
    PERFORM CKUTE-AKTAR VARYING N FROM 1 BY 1
       UNTIL N > VERIUZ (S).
CKUTE-AKTAR.
    MOVE C1X (N) TO CIKTIT (R).
    ADD 1 TO R.
ANAHTARKONTROL-CIK.
    EXIT.
*    ***************************************************************
*    *                                                             *
*    *      THIS PROCEDURE ACCESSES TO THE USER FILE AND THEN      *
*    *    THE RETRIEVED DATA VALUES ARE TRANSFERED TO WORKAREA     *
*    *                                                             *
*    ***************************************************************
 KUTUKTEN-CEK SECTION 2054.
 KUTUKTENCEK.
    PERFORM DAGITIM-FONKSIYONU-AL.
    MOVE KUTUKADI TO GENELKADI.
```

```
    OPEN INPUT GENEL,OUTPUT CIKTIKUT.
    IF SAKLAYIN = 1 OPEN INPUT ECIKTI.
    PERFORM ANAHTAR-DONUSTUR VARYING S FROM 1 BY 1
      UNTIL S > ANAHSAY.
    MOVE 1 TO K.
    IF SAKLAYIN = 1
      CLOSE ECIKTI DISMISS
      PERFORM ECIKTIPOZBUL VARYING I FROM 1 BY 1
        UNTIL I > VERISAYE
      MOVE 0 TO SAKLAYIN.
    PERFORM CIKTIPOZBUL VARYING I FROM 1 BY 1 UNTIL I > VERISAYC.
    CLOSE CIKTIKUT DISMISS.
    CLOSE GENEL LOCK.
    SUBTRACT 1 FROM K GIVING VERISAYE.
    GO TO KUTUKTENCEK-CIK.
DAGITIM-FONKSIYONU-AL.
    MOVE KUTUKADI TO KUTUK-ADI-7.
    READ SISK07 KEY KUTUK-ADI-7 INVALID KEY MOVE 7 TO HATA-KODU
                                PERFORM SISK-HATA.
    MOVE BOLEN TO I.
    PERFORM NUZBUL UNTIL I < 1.
NUZBUL.
    DIVIDE I BY 10 GIVING I.
    ADD 1 TO NL.
ANAHTAR-DONUSTUR.
    MOVE ACIKANAH (S) TO TEKANH.
    PERFORM ANAH-DONUSTUR.
    IF D-F-MODUL-ADI = "DF1" PERFORM BOL
    ELSE IF D-F-MODUL-ADI = "DF2" PERFORM KAYDIR
    ELSE IF D-F-MODUL-ADI = "DF3" PERFORM KATLA.
    MOVE SONUC TO GENANAH.
    PERFORM TUTANAK-OKU.
```

```
DIZITOPLA.
    IF NUM (K) NOT < 10 PERFORM SAYIAL
       MOVE 1 TO SWT1
    ELSE COMPUTE NUM (K) = NUM (K) + SWT1
         MOVE 0 TO SWT1
         IF NUM (K) NOT < 10 PERFORM SAYIAL.
    SUBTRACT 1 FROM K.
SAYIAL.
    COMPUTE NX = NUM (K) / 10.
    COMPUTE NN = NUM (K) - NX * 10.
    IF NN = 0 COMPUTE NUM (K) = 0 + SWT1
       ELSE   COMPUTE NUM (K) = NN + SWT1.
ECIKTIPOZBUL.
    SUBTRACT 1 FROM I GIVING L.
    IF CVERISAKLA (I) = 1
       MOVE CVERIADI (I) TO CVERIADI (K)
       MOVE CVERIUZ  (I) TO CVERIUZ  (K)
    IF I = 1 MOVE 1 TO CVERIBAS (K)
       ADD 1 TO K
    ELSE ADD CVERIBAS (L),CVERIUZ (L) GIVING CVERIBAS (K)
       ADD 1 TO K.
KUTUKTENCEK-CIK. EXIT.
*    ***************************************************************
*                                                                     *
*     *          THIS PROCEDURE FINDS THE FILE NAME IF IT ISNOT     *
*     *                SPECIFIED EXPLICITLY BY THE USER             *
*     *                                                             *
*    ***************************************************************
KUTUK-ADI-BUL SECTION 2055.
KUTUKADIBUL.
    MOVE 1 TO LL.
ILKVERI.
```

```
     MOVE 1 TO K,J,LK.
     MOVE LL TO J.
     MOVE VERIADI (1) TO VERI-ADI-5.
     READ SISK05 KEY VERI-ADI-5 INVALID KEY MOVE 5 TO HATA-KODU
                                PERFORM SISK-HATA.
     UNSTRING KUTUK-ADLARI INTO KUTUKADI
        FOR 15 POINTER J.
     IF KUTUKADI = SPACES MOVE 4 TO HATA-KODU,PERFORM HATA.
DONGU.
     ADD 1 TO K.
     IF K > VERISAY GO KSON.
     MOVE LK TO J.
     MOVE VERIADI (K) TO VERI-ADI-5.
     READ SISK05 KEY VERI-ADI-5 INVALID KEY PERFORM SISK-HATA.
ERISIM-YOLU-BUL.
K-ARA.
     UNSTRING KUTUK-ADLARI INTO KA FOR 15
              POINTER J.
     IF KUTUKADI = SPACES MOVE 4 TO HATA-KODU,PERFORM HATA.
     IF KUTUKADI = KA MOVE 1 TO LK,GO DONGU.
     IF KUTUKADI > KA ADD 15 TO LK,MOVE LK TO J,GO K-ARA.
     ADD 15 TO LL.
     GO ILKVERI.
KSON. EXIT.
*    ***************************************************************
*    *                                                             *
*    *     THIS PROCEDURE DETERMINES THE ACCESSPATH                *
*    *       SATISFYING THE CONDITIONS                             *
*    *                                                             *
*    ***************************************************************
 ERISIM-YOLU-AL SECTION 2056.
 ERISIMYOLUAL.
```

```
    MOVE 0 TO SON-EYOLU.
    IF ISLEC (I) NOT = 2 AND ISLEC (I) NOT = 4
       MOVE KOSUL-VERI (I) TO VERI-ADI
       MOVE KOSUL-ANAH (I) TO VERI-DEGERI
       PERFORM ERIS
       IF ISLEC (I) NOT = 3
          PERFORM VEYALA VARYING K FROM 1 BY 1
             UNTIL K > ANAHSAY
          GO ERISIM-YOLU-SAPTA
       ELSE GO ERISIM-YOLU-SAPTA.
    MOVE KOSUL-VERI (I) TO VERI-ADI.
    MOVE BOSVERI TO VERI-DEGERI.
    PERFORM ERIS.
    MOVE 0 TO SWT1.
    PERFORM KUCUKSE UNTIL SWT1 = 1.
    IF ISLEC (I) = 4
       MOVE KOSUL-VERI (I) TO VERI-ADI
       MOVE KOSUL-ANAH (I) TO VERI-DEGERI
       PERFORM ERIS
       PERFORM VEYALA VARYING K FROM 1 BY 1 UNTIL K > ANAHSAY.
    GO ERISIM-YOLU-SONU.
ERISIM-YOLU-SAPTA.
    IF ISLEC (I) = 1 GO ERISIM-YOLU-SONU.
    IF ISLEC (I) = 6
       PERFORM OLUMSUZ-YAP VARYING K FROM 1 BY 1
          UNTIL K > ANAHSAY
       GO ERISIM-YOLU-SONU.
    IF ISLEC (I) = 3
       MOVE 0 TO SON-EYOLU.
    MOVE 0 TO SWT1.
    PERFORM BUYUKSE UNTIL SWT1 = 1.
ERISIM-YOLU-SONU.
```

```
    IF I = 1 MOVE SON-EYOLU TO ENSON-EYOLU.
    IF MANTIKSAL-ISLEC = 1
       PERFORM VE   VARYING K FROM 1 BY 1 UNTIL K > ANAHSAY.
    IF MANTIKSAL-ISLEC = 2
       PERFORM VEYA VARYING K FROM 1 BY 1 UNTIL K > ANAHSAY.
    GO TO ERISIMYOLUAL-CIK.
VE.
    IF ENSON-ERISIM-YOLU (K) = 1 AND SON-ERISIM-YOLU (K) = 1
       MOVE 1 TO ENSON-ERISIM-YOLU (K)
    ELSE MOVE 0 TO ENSON-ERISIM-YOLU (K).
VEYA.
    IF ENSON-ERISIM-YOLU (K) = 1 OR  SON-ERISIM-YOLU (K) = 1
       MOVE 1 TO ENSON-ERISIM-YOLU (K).
VEYALA.
    IF ERISIM-YOLU (K) = 1 OR SON-ERISIM-YOLU (K) = 1
       MOVE 1 TO SON-ERISIM-YOLU (K).
YINE-ERIS.
    READ SISK10 NEXT AT END MOVE 1 TO SWT1.
    IF DEVIRME-TUR = "KISMI" PERFORM ERISIM-YOLU-BUL.
    PERFORM GUNLE-SISK10.
GUNLE-SISK10.
    ACCEPT TARIH FROM DATE.
    MOVE TARIH TO SON-KULLANIM-TARIHI.
    ADD 1 TO KULLANIM-SAYISI.
    REWRITE TUTANAK10 INVALID KEY MOVE 10 TO HATA-KODU
       PERFORM SISK-HATA.
ERIS.
    READ SISK10 INVALID KEY PERFORM ERISIM-YOLU-BUL.
KUCUKSE.
    PERFORM YINE-ERIS.
    IF VERI-DEGERI < KOSUL-ANAH (I) AND VERI-DEGERI NOT = SPACES
       PERFORM VEYALA VARYING K FROM 1 BY 1 UNTIL K > ANAHSAY
```

```
     ELSE MOVE 1 TO SWT1.
 BUYUKSE.
     PERFORM YINE-ERIS.
     IF VERI-DEGERI NOT = SPACES
       PERFORM VEYALA VARYING K FROM 1 BY 1 UNTIL K > ANAHSAY
     ELSE MOVE 1 TO SWT1.
 ERISIMYOLUAL-CIK. EXIT.
*    ****************************************************************
*    *                                                              *
*    *          THIS PROCEDURE INITIALIZES SOME VARIABLES           *
*    *                IF THEY IS MORE THAN ONE KEY                  *
*    *                                                              *
*    ****************************************************************
 ANAHTAR-BIRDEN-FAZLA SECTION 2057.
 BIRDENFAZLA.
     MOVE ACIKAN TO ACIKANC.
     MOVE SPACES TO ACIKAN, KX.
     MOVE ANAHSAY TO ANAHSAYC.
     MOVE 0 TO ANAHSAY, SONANAH.
     MOVE 1 TO NX.
     MOVE KUTUKADI TO KUTUK-ADI-11.
     READ SISK11 KEY KUTUK-ADI-11 INVALID KEY MOVE 11 TO HATA-KODU
                                  PERFORM SISK-HATA.
 BIRDENFAZLA-SON. EXIT.
*    ****************************************************************
*    *                                                              *
*    *          THIS PROCEDURE IS THE SET EQUALITY OPTION           *
*    *                OF SET CONDITIONAL RETRIEVAL STATEMENT        *
*    *                                                              *
*    ****************************************************************
 ESIT-ANAHTARLARI-AL SECTION 2058.
 ESITANAHTARLAR.
```

```
           PERFORM ANAHTAR-PARCALA.
           IF KA = KX
              MOVE KA TO KX
              MOVE 0 TO NL
           ELSE
              IF NL = ANAHSAYC
                 ADD 1 TO ANAHSAY
                 MOVE KX TO ACIKANAH (ANAHSAY).
           PERFORM ESIT-MI VARYING I FROM 1 BY 1 UNTIL I > ANAHSAYC.
       ESITANAHTARLAR-SON. EXIT.
      *    ****************************************************************
      *    *                                                              *
      *    *          THIS PROCEDURE IS THE SET INCLUSION OPTION          *
      *    *            OF SET CONDITIONAL RETRIEVAL STATEMENT            *
      *    *                                                              *
      *    ****************************************************************
       ICEREN-ANAHTARLARI-AL SECTION 2059.
       ICERENANAHTARLAR.
           PERFORM ANAHTAR-PARCALA.
           IF KA = KX
              MOVE KA TO KX
              MOVE 0 TO NL
           ELSE
              IF NL NOT < ANAHSAYC
                 ADD 1 TO ANAHSAY
                 MOVE KX TO ACIKANAH (ANAHSAY).
           PERFORM ESIT-MI VARYING I FROM 1 BY 1 UNTIL I > ANAHSAYC.
       ICERENANAHTARLAR-SON. EXIT.
      *    ****************************************************************
      *    *                                                              *
      *    *          THIS PROCEDURE IS THE SET INEQUALITY OPTION         *
      *    *            OF SET CONDITIONAL RETRIEVAL STATEMENT            *
```

```
*  *                                                                *
*    ****************************************************************
 DISINDAKI-ANAHTARLARI-AL SECTION 2060.
 DISINDAKIANAHTARLAR.
     PERFORM ANAHTAR-PARCALA.
     IF KA = KX
        IF NL > 0 GO DISINDAKIANAHTARLAR-SON
        ELSE NEXT SENTENCE
     ELSE
        IF NL = 0 AND SWT1 = 1
        ADD 1 TO ANAHSAY
           MOVE KX TO ACIKANAH (ANAHSAY).
     IF KA NOT = KX
        MOVE KA TO KX
        MOVE 0 TO NL.
     PERFORM ESIT-MI VARYING I FROM 1 BY 1 UNTIL I > ANAHSAYC.
     MOVE 1 TO SWT1.
 DISINDAKIANAHTARLAR-SON. EXIT.
*    ****************************************************************
*    *                                                              *
*    *                  HASHING FUNCTION DF1 :                      *
*    *                        D I V I D E                           *
*    *                                                              *
*    ****************************************************************
 BOL SECTION 3051.
 BOL-BAS.
     MOVE DL TO M.
     PERFORM CARP VARYING I FROM 1 BY 1 UNTIL I > DL.
     DIVIDE SAYAC BY BOLEN GIVING SAYAC REMAINDER SONUC.
     IF SONUC = 0 ADD 1 TO SONUC.
 CARP.
     COMPUTE SAYAC = RAKAMLAR (I) * 10 ** (M - I) + SAYAC.
```

```
BOL-SON. EXIT.
*    ****************************************************************
*    *                                                              *
*    *                  HASHING FUNCTION DF2 :                      *
*    *                        S H I F T                             *
*    *                                                              *
*    ****************************************************************
 KAYDIR SECTION 3052.
 KAYDIR-BAS.
     MOVE 0 TO NU.
     COMPUTE KL = - NL.
     PERFORM DIZITOP VARYING I FROM 1 BY 1 UNTIL I > NL.
     MOVE NL TO K.
     MOVE 0 TO SWT1, NUMMS.
     PERFORM KATLAKAYDIR.
     GO TO KAYDIR-SON.
 DIZITOP.
     COMPUTE RAKAM = DL - (I - 1).
     COMPUTE   J = NL - (I - 1).
     PERFORM TOB VARYING K FROM RAKAM BY KL  UNTIL K < 1.
 TOB.
     ADD RAKAMLAR (K) TO NUM(J).
 KATLAKAYDIR.
     PERFORM DIZITOPLA UNTIL K = 0.
     MOVE 0 TO RAKAM-LAR.
     PERFORM AKTAR VARYING I FROM 1 BY 1 UNTIL I > NL.
     MOVE NL TO M.
     PERFORM CARP VARYING I FROM 1 BY 1 UNTIL I > NL.
     COMPUTE SONUC = SAYAC * (BOLEN / 10 ** NL).
     IF SONUC = 0 ADD 1 TO  ONUC.
 AKTAR.
     MOVE NUM (I) TO RAKAMLAR (I).
```

```
 CARP.
     COMPUTE SAYAC = RAKAMLAR (I) * 10 ** (M - I) + SAYAC.
 KAYDIR-SON. EXIT.
*    ****************************************************************
*    *                                                              *
*    *                  HASHING FUNCTION DF3 :                      *
*    *                        F O L D                               *
*    *                                                              *
*    ****************************************************************
 KATLA SECTION 3053.
 KATLA-BAS.
     MOVE DL TO K.
     MOVE 2 TO NL, J.
     MOVE 1 TO SWT1.
     MOVE 0 TO NU.
     PERFORM TEKKATLA UNTIL K < 1.
     SUBTRACT 1 FROM J.
     SUBTRACT 1 FROM DL GIVING K.
     MOVE 0 TO SWT1.
     PERFORM TEKKATLA UNTIL K < 1.
     MOVE NL TO K.
     MOVE 0 TO SWT1.
     PERFORM KATLAKAYDIR.
     GO TO KATLA-SON.
 TEKKATLA.
     ADD RAKAMLAR (K) TO NUM (J).
     IF SWT1 = 1 SUBTRACT 3 FROM K
          MOVE 0 TO SWT1
       ELSE SUBTRACT 1 FROM K
          MOVE 1 TO SWT1.
 KATLAKAYDIR.
     PERFORM DIZITOPLA UNTIL K = 0.
```

```
    MOVE 0 TO RAKAM-LAR.
    PERFORM AKTAR VARYING I FROM 1 BY 1 UNTIL I > NL.
    MOVE NL TO M.
    PERFORM CARP VARYING I FROM 1 BY 1 UNTIL I > NL.
    COMPUTE SONUC = SAYAC * (BOLEN / 10 ** NL).
    IF SONUC = 0 ADD 1 TO SONUC.
AKTAR.
    MOVE NUM (I) TO RAKAMLAR (I).
CARP.
    COMPUTE SAYAC = RAKAMLAR (I) * 10 ** (M - I) + SAYAC.
KATLA-SON. EXIT.
*    ***************************************************************
*    *                                                             *
*    *          THIS PROCEDURE READS A RECORD OF USER FILE         *
*    *          AND STORES IT AT THE WORKAREA                      *
*    *                                                             *
*    ***************************************************************
TUTANAK-OKU SECTION 3054.
TUTANAKOKU-BAS.
    READ GENEL INVALID KEY PERFORM KOD-KONTROL.
    MOVE SPACES TO CIKTITUT , C1.
    MOVE 1 TO R.
    UNSTRING GENTUT INTO C1 FOR ANAH-UZUNLUGU WITH POINTER R.
    MOVE ACIKANAH (S) TO C2.
    IF C1 NOT = C2
       ADD 1 TO SONUC
       MOVE SONUC TO GENANAH
       GO TO TUTANAKOKU-BAS.
    MOVE 1 TO R.
    IF SAKLAYIN = 1 PERFORM ECIKTIDAN-AKTAR.
    PERFORM PARCALAMA VARYING I FROM 1 BY 1 UNTIL I > VERISAYC.
    WRITE CIKTITUT.
```

```
    GO TO TUTANAKOKU-SON.
ECIKTIDAN-AKTAR.
    READ ECIKTI AT END MOVE 1 TO SWT1.
    PERFORM ECIKTIDAN-AL VARYING I FROM 1 BY 1
       UNTIL I > VERISAYE.
ECIKTIDAN-AL.
    IF CVERISAKLA (I) = 1
       MOVE CVERIBAS (I) TO K
       PERFORM CIKTIKUTE-AKTAR VARYING LL FROM K BY 1
          UNTIL LL > CVERIUZ (I).
CIKTIKUTE-AKTAR.
    MOVE ECIK (LL) TO CIKTIT (R).
    ADD 1 TO R.
PARCALAMA.
    MOVE SPACES TO C1.
    MOVE VERIBAS (I) TO N.
    UNSTRING GENTUI INTO C1 FOR VERIUZ (I)
       POINTER N.
    PERFORM CKUTE-AKTAR VARYING N FROM 1 BY 1
       UNTIL N > VERIUZ (I).
CKUTE-AKTAR.
    MOVE C1X (N) TO CIKTIT (R).
    ADD 1 TO R.
TUTANAKOKU-SON. EXIT.
*    ****************************************************************
*    *                                                              *
*    *          THIS PROCEDURE CONVERTS THE SYMBOLIC KEY OF THE     *
*    *          USER FILE TO A NUMERIC KEY VALUE                    *
*    *                                                              *
*    ****************************************************************
ANAH-DONUSTUR SECTION 3055.
ANAHDONUSTUR.
```

```
           MOVE 0 TO L.
           INSPECT TEKANH TALLYING L FOR CHARACTERS BEFORE INITIAL " ".
           MOVE 0 TO DL,RAKAM-LAR.
           PERFORM CEVIR VARYING K FROM 1 BY 1 UNTIL K > L.
           SUBTRACT 15 FROM DL GIVING LL.
           MOVE 0 TO SAYAC.
           IF DL > 15 MOVE 0 TO NU
              PERFORM EKLE VARYING K FROM 1 BY 1 UNTIL K > 15
              MOVE 15 TO M
              PERFORM DIZITOPLA.
           GO ANAHDONUSTUR-SON.
       CEVIR.
           MOVE K TO I.
           ADD 1 TO DL.
           MOVE 0 TO RAKAM.
           UNSTRING TEKANH INTO KEY1 FOR 1 WITH POINTER I.
           INSPECT C37 TALLYING RAKAM FOR CHARACTERS BEFORE INITIAL
              KEY1.
           IF RAKAM < 10 MOVE RAKAM TO RAKAMLAR (DL)
           ELSE COMPUTE RAKAMLAR (DL) = RAKAM / 10
              ADD 1 , DL GIVING KL
              COMPUTE RAKAMLAR (KL) = RAKAM - RAKAMLAR (DL) * 10
              ADD 1 TO DL.
       EKLE.
           ADD K TO LL.
           PERFORM EKLET VARYING I FROM LL BY -15 UNTIL I < 0.
       EKLET.
           COMPUTE NUM (K) = NUM (K) + RAKAMLAR (I).
       ANAHDONUSTUR-SON. EXIT.
      *    ************************************************************
      *    *                                                          *
      *    *           THIS PROCEDURE DIVIDES THE KEY DATA ITEMS      *
```

```
*    *                  IF THEY ARE MORE THAN ONE                    *
*    *                                                               *
*    *****************************************************************
 ANAHTAR-PARCALA SECTION 3056.
 ANAHTARPARCALA.
     MOVE SPACES TO TEKANH,DEGERI,KA.
     MOVE NX TO LL.
     UNSTRING ANAHTAR-LISTESI  INTO TEKANH FOR ANAH-UZUNLUGU
        WITH POINTER LL.
     IF TEKANH = SPACES MOVE 1 TO SONANAH GO ANAHTARPARCALA-SON.
     ADD ANAH-UZUNLUGU TO NX.
     MOVE J TO LL.
     UNSTRING TEKANH INTO DEGERI FOR LK WITH POINTER LL.
     MOVE VERIBAS (1) TO S.
     UNSTRING TEKANH INTO KA FOR VERIUZ (1)
        WITH POINTER S.
 ANAHTARPARCALA-SON. EXIT.
```