NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ARCHIEVE PROGRAM

COM 400 PROJECT

ŞEFKİ KOLOZALİ	**(20010333)**
NEBİLE KIRMIZILAR	**(20000325)**

SUBMİTTED TO: ÜMİT SOYER

NICOSIA 2005

ACKNOWLEDGEMENT

We would like to thank to our supervisors Umit Soyer , without whom this project would have not been possible. Whose words of encouragement kept us doing our project. His invaluable knowledge for the project has made our taking keen interest in our project.

Specially we would like to thank Hasan Yucel who helped us so much in doing our project. We would also like to thank Caner Cakir for his helped, he was always near us and helped us. Also thank for all our friend to be belive us and give as moral: Yazen Al Shawneh , Sule Akkurt ,Mustafa Sevki Hizal ,Duygu Ozsen and Ahmet Unler. Being with them make our time full of fun. They encouraged us allot in completing our project. We want to thank them as they contributed their time and provided very helpful suggestions.

Finally, we want to thank our family, especially our parents. Without their endless support and love for us, we would never achieve our current position.

ABSTRACT

The archive system is the principles of organization and description. It use for to find our documents more easy than looking step by step all the documents. In our project we did an archive program to fine documents and images more easy without looking anywhere just write name of what you are looking and search so it will be more easy to find what we are looking. With out move your place you can find what ever you are looking. But all of this making with controlling. But we have to know how we will fine. Research is work that tries to discover facts about something. if there is resemblance between two things there similar to each other. if you research something ,you try to discover facts about it.

NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ARCHIEVE PROGRAM

COM 400 PROJECT

ŞEFKİ KOLOZALİ (20010333)

NEBİLE KIRMIZILAR (20000325)

SUBMİTTED TO: ÜMİT SOYER

NICOSIA 2005

ACKNOWLEDGEMENT

We would like to thank to our supervisors Umit Soyer , without whom this project would have not been possible. Whose words of encouragement kept us doing our project. His invaluable knowledge for the project has made our taking keen interest in our project.

Specially we would like to thank Hasan Yucel who helped us so much in doing our project. We would also like to thank Caner Cakir for his helped, he was always near us and helped us. Also thank for all our friend to be belive us and give as moral: Yazen Al Shawneh , Sule Akkurt ,Mustafa Sevki Hizal ,Duygu Ozsen and Ahmet Unler. Being with them make our time full of fun. They encouraged us allot in completing our project. We want to thank them as they contributed their time and provided very helpful suggestions.

Finally, we want to thank our family, especially our parents. Without their endless support and love for us, we would never achieve our current position.

ABSTRACT

The archive system is the principles of organization and description. It use for to find our documents more easy than looking step by step all the documents. In our project we did an archive program to fine documents and images more easy without looking anywhere just write name of what you are looking and search so it will be more easy to find what we are looking. With out move your place you can find what ever you are looking. But all of this making with controlling. But we have to know how we will fine. Research is work that tries to discover facts about something. if there is resemblance between two things there similar to each other. if you research something ,you try to discover facts about it.

TABLE OF CONTENTS

ACKNOWLEDGEMENT i
ABSTRACT ii
TABLE OF CONTENTS iii
INTRODUCTION viii

CHAPTER 1 1

THE ARCHITECTURE OF DATABASE APPLICATIONS 1

1.1 ARCHITECTURE OF DATABASE 1

1.1.1 RELATIONAL DATABASE CONCEPTS 1

1.1.1.1 DATABASE TABLES 1

1.1.1.1.1 CHARACTERISTICS OF RELATIONAL TABLES 2

1.1.1.1.2 WHAT DO YOU KEEP IN EACH TABLE? 3

1.1.1.2 RELATIONS BETWEEN TABLES 3

1.1.1.2.1 TABLES HAVE A PRIMARY KEY 4

1.1.1.2.1.1 EXAMPLES 4

1.1.1.2.2 TABLES ARE RELATED BY FOREIGN KEYS 4

1.1.1.2.2.1 EXAMPLE 5

1.1.1.3 OTHER DATABASE OBJECTS 5

1.1.2 THE PIECES OF A DATABASE SYSTEM 7

1.1.3 HOW THE PIECES FIT TOGETHER 9

1.1.3.1 PERSONAL APPLICATIONS AND EMBEDDED DATABASES 9

1.1.3.2 CLIENT/SERVER APPL. AND MULTI-USER DATABA 10

1.1.4 MULTI-TIER COMPUTING ARCHITECTURE 11

1.1.5 USING MULTIPLE DATABASES 12

1.1.5.1 RUNNING DATABASES ON A SINGLE DATABASE SERVER 12

1.1.5.2 ACCESSING DATA IN OTHER DATABASES 13

1.2 ABOUT DBASE AND PARADOX 13

1.2.1 DBASE IV TABLE SPECIFICATION 13

1.2.2 DBASE V TABLE SPECIFICATIONS 14

1.2.3 DBASE FIELD TYPES 14

1.2.4 PARADOX STANDARD TABLE SPECIFICATIONS 15

1.2.5 PARADOX 5 TABLE SPECIFICATIONS 16

1.2.6 PARADOX 7 AND ABOVE TABLE SPECIFICATIONS 17

1.2.7 PARADOX FIELD TYPES 17

1.2.8 ABOUT OUR DATABASE **19**

CHAPTER 2 22

OPERATION SYSTEM 22

2.1 OPERATING SYSTEM 22

2.2 OPERATING SYSTEM TYPES 22

2.2.1 MULTI-USER 23

2.2.2 MULTIPROCESSING 23

2.2.3 MULTITASKING 23

2.2.4 MULTITHREADING 23

2.3 INTRODUCTION TO UNIX: 24

2.3.1 OBJECTIVES 24

2.3.2 WHAT IS AN OPERATING SYSTEM? 24

2.3.3 A BRIEF HISTORY OF UNIX 26

2.3.4 ARCHITECTURE OF THE LINUX OPERATING SYSTEM 29

2.4 AN INTRODUCTION TO LINUX 31

2.4.1 WHAT IS LINUX? 31

2.4.2 WHY USE LINUX? 32

2.4.2.1 CONFIGURABILITY 32

2.4.2.2 CONVENIENCE 33

2.4.2.3 STABILITY 33
2.4.2.4 COMMUNITY 33
2.4.2.5 FREEDOM 34
2.5 THE DIFFERENCES BETWEEN LINUX AND OTHER UNIX 34
CHAPTER 3 35
DELPHI.. *35*
3.1.INTRODUCTION.. *35*
3.2. ASYNCHRONOUS INTERACTION *37*
3.3 ANONYMITY *40*
3.4 MODERATION AND FACILITATION *43*
3.5 STRUCTURE *44*
3.5.1 The Policy Delphi *45*
3.5.2 The Trend Model *47*
3.6 ANALYSIS.. *50*
3.6.1 Scaling Methods *50*
3.6.2 Structural Modeling.. *53*
3.7 DELPHI, EXPERT SYS., GDSS, AND COLLABORATIVE SYSTEMS..... 55
3.7.1 The Deductive Level of Disagreements *57*
3.7.2 The Inductive Level of Disagreements *58*
3.7.3 Goal and Value Disagreements *59*
3.8 CONCLUSION *60*
CHAPTER 4 61
ARCHIVE SYSTEM 61
4.1 Basic concepts and principles of archives and records management 62
4.1.1 The Inter-relatedness of records 62
4.1.2 The central importance of context.. 62
4.1.3 The function of records as evidence 63
4.1.3.1 Recordness requires that records are: 63
4.1.3.1.1 Complete.. 64

4.1.3.1.2 'Fixed' 64

4.1.3.1.3 Organic 64

4.1.3.1.4 Contextual 64

4.1.3.1.5 Authoritative/'Official' 65

4.1.3.1.6 Unique 65

4.1.3.1.6.1. The principle of provenance 65

4.1.3.1.6.2. The principle of original order 65

4.1.3.1.6.3. The chain of responsible custody 67

4.1.3.1.6.3.1 What goes on in the archives? 67

4.1.3.1.6.3.2 CADSS 67

4.1.3.1.6.3.3 CONTROL 68

4.1.3.1.6.3.4 ACCESSIBILITY 69

4.1.3.1.7 DISPOSAL 71

4.1.3.1.8 STORAGE 72

4.1.3.1.9 Check Answer 73

4.1.3.1.10 SUSTAIN 75

4.1.3.1.11 Recordkeeping regimes 77

4.1.3.1.12 Where we work: The recordkeeping context 77

4.1.3.1.12.1 Public sector placements 77

4.1.3.1.12.2 Private sector placements 78

4.1.3.1.13 Type and functional emphasis 78

4.1.3.1.14 Accountable recordkeeping regimes at macro level. 79

4.1.3.1.15 Recordkeeping rewards 82

4.1.3.1.16 Recordkeeping and related professions 82

4.1.3.1.16.1 What's the difference? 82

4.1.3.1.17 Recordkeeping and specialist recordkeepers 83

CHAPTER 5 .... **84**

ARCHIVE PROGRAM 84

CONCLUSION 157

REFERNCES……………………………………………………………………………157

INTRODUCTION

Archive system is main base registration function of office management for extension. For correct registration of incoming and outgoing files or documents you need always true programming system. Our program is using computer for registration sending and receiving documents from selected office. Documents are enter to the Archive system with their attachments. Any one can easily reach all of enclosed file or documents.

Our first chapter is all about explaining Architecture of database systems. About relational database's concepts, the pieces of database systems, how the pieces fit together, multi-tier computing architecture and multiple databases, dbase and paradox.

Our second chapter is all about giving details of operation systems. Which are Unix, Linux and the differential of Unix and Linux.

Our third chapter is all about Delphi. Introduction of Delphi, asynchronous interaction ,anonymity and so on.

Our fourth chapter we explained about the archive systems. Basic concepts and principles of archives and how its work.

Our last fifth chapter is about the our program archive program how its wo.how we can use it and so on.

CHAPTER 1

The Architecture of Database Applications

1.1 Architecture of database

1. Relational database concepts
2. The pieces of a database system
3. How the pieces fit together
4. Multi-tier computing architecture
5. Using multiple databases
6. About dbase

1.1.1 Relational database concepts

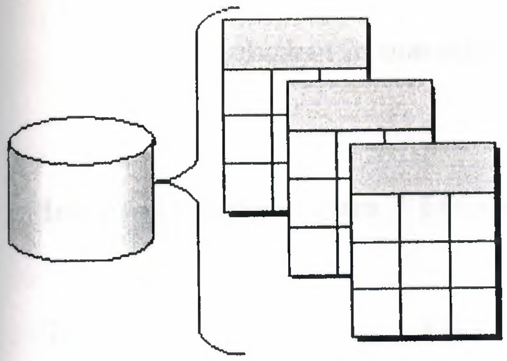
A relational database-management system (RDBMS) is a system for storing and retrieving data, in which the data is organized into interrelated tables.

SQL Anywhere Studio provides two relational database systems. Adaptive Server Anywhere is the primary, full featured RDBMS, with a multitude of uses, from a network database server hosting many clients to a compact embedded database. UltraLite is a small-footprint relational database. The UltraLite deployment technology allows you to use Adaptive Server Anywhere features on even the smallest of devices.

1.1.1.1 Database tables
1.1.1.2 Relations between tables
1.1.1.3 Other database objects

1.1.1.1 Database tables

In a relational database, all data is held in tables, which are made up of rows and columns.



Each table has one or more columns, and each column is assigned a specific data type, such as an integer, a sequence of characters (for text), or a date. Each row in the table has a single value for each column.

For example, a table containing employee information may look as follows:

emp_ID	emp_lname	emp_fname	emp_phone
10057	Huong	Zhang	1096
10693	Donaldson	Anne	7821

1.1.1.1.1 Characteristics of relational tables

The tables of a relational database have some important characteristics:

- There is no significance to the order of the columns or rows.
- Each row contains one and only one value for each column, or contains NULL, which indicates that there is no value for that column.
- All values for a given column have the same data type.

  The following table lists some of the formal and informal relational database terms describing tables and their contents, together with their

equivalent in non-relational databases. This manual uses the informal terms.

Informal relational term	Formal relational term	Non-relational term
Table	Relation	File
Column	Attribute	Field
Row	Tuple	Record

1.1.1.1.2 What do you keep in each table?

Each table in the database should hold information about a specific thing, such as employees, products, or customers.

By designing a database this way, you can set up a structure that eliminates redundancy and the possible inconsistencies caused by redundancy. For example, both the sales and accounts payable departments might enter and look up information about customers. In a relational database, the information about customers is stored only once, in a table that both departments can access.

1.1.1.2 Relations between tables

You use primary keys and foreign keys to describe relationships between the information in different tables. Primary keys identify each row in a table uniquely, and foreign keys define the relationships between rows in different tables.

Primary keys and foreign keys let you use relational databases to hold information in an efficient manner, without redundancy.

1.1.1.2.1 Tables have a primary key

1.1.1.2.2 Tables are related by foreign keys

1.1.1.2.1 Tables have a primary key

Each table in a relational database should have a primary key. The primary key is a column, or set of columns, that uniquely identifies each row. No two rows in a table may have the same primary key value

.

1.1.1.2.1.1 Examples

In the sample database, the employee table stores information about employees. It has a primary key column named emp_id, which holds a unique ID number assigned to each employee. A single column holding an ID number is a common way to assign primary keys, and has advantages over names and other identifiers that may not always be unique.

A more complex primary key can be seen in the sales_order_items table of the sample database. The table holds information about individual items on orders from the company, and has the following columns:

id	An order number, identifying the order the item is part of.
line_id	A line number, identifying each item on any order.
prod_id	A product ID, identifying the product being ordered.
quantity	A quantity, displaying how many items were ordered.
ship_date	A ship date, displaying when the order was shipped.

A particular sales order item is identified by the order it is part of, and by a line number on the order. These two numbers are stored in the id and line_id columns. Items may share a single id value (corresponding to an order for more than one item) or they may share a line_id number (all first items on different orders have a line_id of 1). No two items share both values, and so the primary key is made up of these two columns.

1.1.1.2.2 Tables are related by foreign keys

The information in one table is related to that in other tables by foreign keys.

1.1.1.2.2.1 Example

The sample database has one table holding employee information and one table holding department information. The department table has the following columns:

- dept_id An ID number for the department. This is the primary key for the table.
- dept_name The name of the department.
- dept_head_id The employee ID for the department manager.

To find the name of a particular employee's department, there is no need to put the name of the employee's department into the employee table. Instead, the employee table contains a column holding a number that matches one of the dept_id values in the department column.

The dept_id column in the employee table is called a foreign key to the department table. A foreign key references a particular row in the table containing the corresponding primary key.

In this example, the employee table (which contains the foreign key in the relationship) is called the foreign table or referencing table. The department table (which contains the referenced primary key) is called the primary table or the referenced table.

1.1.1.3 Other database objects

There is more to a relational database than simply a set of related tables. You will also find the following objects in a relational database:

- Indexes Indexes allow quick lookup of information. Conceptually, an index in a database is like an index in a book. In a book, the index relates each indexed term to the page or pages on which that word appears. In a database, the index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored.

Indexes are an important design element for high performance. You must usually create indexes explicitly, but indexes for primary and foreign keys and for unique columns are created automatically. Once created, the use of indexes is transparent to the user.

- Views Views are computed tables, or virtual tables. They look like tables to client applications, but they do not hold data. Instead, whenever they are accessed, the information in them is computed from the underlying tables.

  The tables that actually hold the information are sometimes called base tables to distinguish them from views. A view is defined with a SQL query on base tables or other views.

- Stored procedures and triggers These are routines held in the database itself that act on the information in the database.

  You can create and name your own stored procedures to execute specific database queries and to perform other database tasks. Stored procedures can take parameters. For example, you might create a stored procedure that returns the names of all customers who have spent more than the amount that you specify as a parameter in the call to the procedure.

  A trigger is a special stored procedure that automatically fires whenever a user updates, deletes, or inserts data, depending on how you define the trigger. You associate a trigger with a table or columns within a table. Triggers are useful for automatically maintaining business rules in a database.

- Users and groups Each user of a database has a user ID and password. You can set permissions for each user so that confidential information is kept private and users are prevented from making unauthorized changes. Users can be assigned to groups in order to make the administration of permissions easier.

- Java objects You can install Java classes into the database. Java classes provide a powerful way of building logic into your

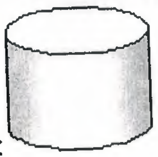
database, and a special class of user-defined data types for holding information.

All of these items together make up a relational database-management system (RDBMS); a system for storing and retrieving data, in which the data is organized into interrelated tables.

1.1.2 The pieces of a database system

Relational database management systems contain the following pieces:

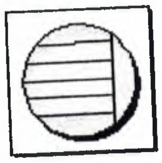
A database   Data is stored in a database. In diagrams in the documentation, a database is indicated by a cylinder:



An Adaptive Server Anywhere database is a file, usually with an extension of .db. Adaptive Server Anywhere includes a sample database for you to work with: this is the file asademo.db in your Adaptive Server Anywhere installation directory.

A database server   The database server manages the database. No other application addresses the database file directly; they all communicate with the database server.

In the documentation, a database server is indicated as follows:



Adaptive Server Anywhere provides two versions of its database server: the personal database server and the network database server. In addition to the features of the personal server, the network database server also supports client/server communications across a network, while the personal database server can accept connections only from applications running on the same machine. The request-processing engine is identical in both servers.

A programming interface   Applications communicate with the database server using a programming interface. You can use ODBC, JDBC, OLE DB, Sybase Open Client, or Embedded SQL.

Many application development tools provide their own programming environment that hides the details of the underlying interface. For example, if you develop an application using Sybase PowerBuilder, you never have to make an ODBC function call. Nevertheless, behind the scenes each of these tools is using one of the programming interfaces.

The programming interface provides a library of function calls for communicating with the database. For ODBC and JDBC, the library is commonly called a **driver**. The library is typically provided as a shared library on UNIX operating systems or a dynamic link library (DLL) on PC operating systems. The JDBC interface uses the Sybase jConnect driver, which is a zip file of compiled Java classes.

A client application   Client applications use one of the programming interfaces to communicate with the database server.

If you develop an application using a rapid application development (RAD) tool such as Sybase PowerBuilder, you may find that the tool provides its own methods for communicating with database servers, and hides the details of the language interface. Nevertheless, all applications use one of the supported interfaces.

In diagrams in the documentation, a client application is indicated by the following icon:

UltraLite database servers are custom-generated for each UltraLite application, and are part of the application itself. An UltraLite application together with its database server is indicated as follows:

1.1.3 How the pieces fit together

Database applications can connect to a database server located on the same machine as the application itself, or in the case of the network database server, on a different machine. In addition, with Adaptive Server Anywhere you can build distributed databases, with physically distinct databases on different machines sharing data.

1.1.3.1 Personal applications and embedded databases
1.1.3.2 Client/server applications and multi-user databases

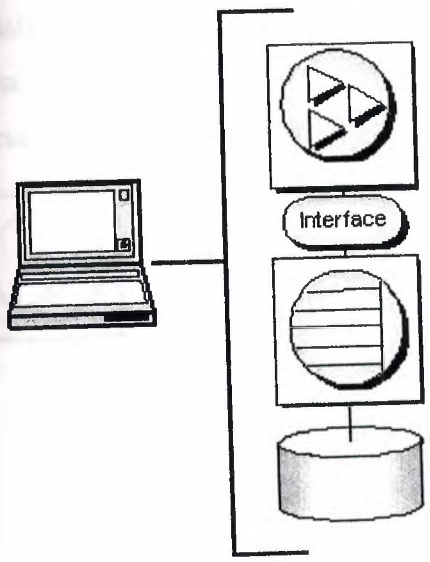
1.1.3.1 Personal applications and embedded databases

You can use Adaptive Server Anywhere to build a complete application and database on a single computer. In the simplest arrangement, this is a standalone application or personal application: it is self-contained with no connection to other databases. In this case, the database server and the database can be started by the client application, and it is common to refer to the database as an embedded database. As far as the end user is concerned, the database is a part of the application. When a database server is used as an embedded database, it is sometimes called a database engine.

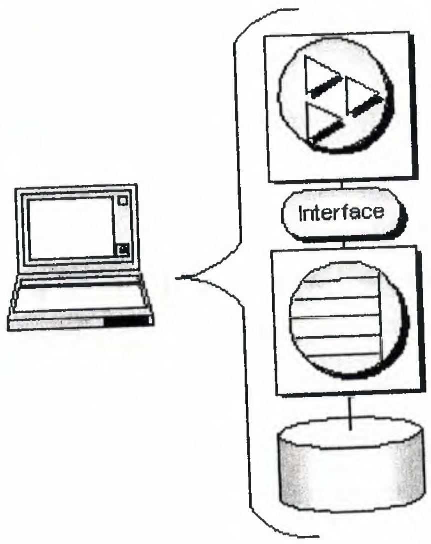
Many relational database management systems require experienced staff for administration. A characteristic of embedded databases is the ability to run entirely without administration.

The Adaptive Server Anywhere personal database server is generally used for standalone applications. A client application connects through a programming interface to a database server running on the same machine:

Standalone applications have the following architecture, with a client application connecting through a programming interface to a database server running on the same machine:



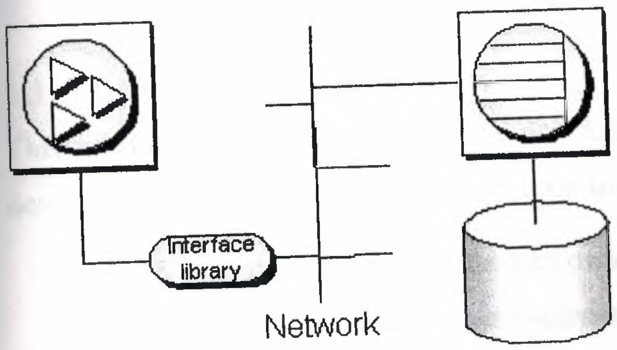
The Adaptive Server Anywhere personal database server is generally used for standalone applications, although you can also use applications on the same machine as the network server.



1.1.3.2 Client/server applications and multi-user databases

You can use Adaptive Server Anywhere to build an installation with many applications running on different machines, connected over a network to a single

database server running on a separate machine. This is a client/server or multi-user database environment, and has the following architecture. The interface library is located on each client machine.



In this case, the database server is the Adaptive Server Anywhere network database server, which supports network communications.

For a client application to work in a client/server environment, you need only identify the server to which it should connect.

For a client application to work in a client/server environment, you need only identify the server to which it should connect.

1.1.4 Multi-tier computing architecture

In multi-tier computing, application logic is held in an application server, such as Sybase EAServer, which sits between the database server and the client applications. In many situations, a single application server may access multiple databases in addition to non-relational data stores. In the Internet case, client applications are browser-based, and the application server is generally a Web server extension.

Sybase EAServer stores application logic in the form of components, and makes these components available to client applications. The components may be PowerBuilder components, Java beans, or COM components.

Application servers can also provide transaction logic to their client applications—guaranteeing that sets of operations are executed atomically across multiple databases. Adaptive Server Anywhere is well-suited to multi-tier computing, and can participate in distributed transactions coordinated by Microsoft Distributed Transaction Coordinator.

Both Sybase Enterprise Application Server and Microsoft Transaction Server use DTC to provide transaction services to their client applications.

1.1.5 Using multiple databases

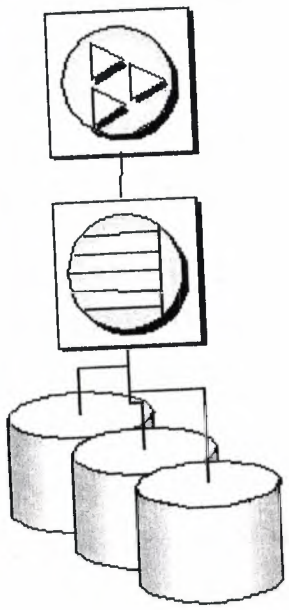
This section describes extensions to the Adaptive Server Anywhere architecture described above for the case where you want to use more than one database.

1.1.5.1 Running multiple databases on a single database server

1.1.5.2 Accessing data in other databases

1.1.5.1 Running multiple databases on a single database server

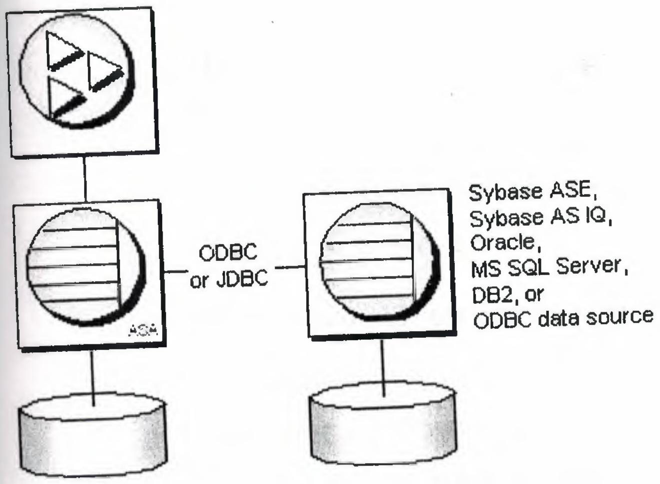
Both the Adaptive Server Anywhere personal database server and the network database server can manage several databases simultaneously. Each connection from an application must be to a single database, but an application can use separate connections to different databases, or a set of applications can work on different databases, all through the same database server.



Databases can be started when the database server is started, or by connecting to a database using the DatabaseFile connection parameter.

1.1.5.2 Accessing data in other databases

You can access databases on multiple database servers, or even on the same server, using the Adaptive Server Anywhere Remote Data Access features. The application is still connected to a single database as in the architecture diagrams above, but by defining remote servers, you can use proxy tables that exist on the remote database as if they were in the database to which you are connected.



1.2 About Dbase And Paradox

1.2.1 dBASE IV Table Specification

The dBASE IV table format was introduced in dBASE IV for DOS. Following are the specifications for dBASE IV tables.

2GB file size.

Two billion records per file.

A maximum of 255 fields per record.

Maintained indexes can have up to 47 indexes per file. Each index can be created using field expressions of virtually any combination, including conditional expressions of up to 255 characters per expression that result in an index of up to 100 bytes.

Unlimited nonmaintained indexes can be stored on disk. You can use up to 47 of them simultaneously.

1.2.2 dBase V Table Specifications

The dBASE V table format was introduced in dBASE V for Windows. Following are the specifications for dBASEA V tables.

- Up to one billion records per file.
- A maximum of 1,024 fields per record.
- Up to 32,767 bytes per record.
- Unlimited nonmaintained indexes can be stored on disk. You can use up to 47 of them simultaneously.
- Up to 10 master index files open per database. Each master index can have up to 47 indexes.
- Maintained indexes can have up to 47 indexes per file. Each index can be created using field expressions of virtually any combination, including conditional expressions of up to 255 characters per expression that result in an index of up to 100 bytes.

1.2.3 dBASE Field Types

Character (C)

dBASE III+, IV, and V field type that can contain up to 254 characters (including blank spaces). This field is similar to the Paradox Alpha field type.

Date (D)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V. dBASE tables can store dates from January 1, 100, to December 31, 9999. Paradox 5 tables can store from 12/31/9999 B.C. to 12/31/9999 A.D.

Float (F)

dBASE IV, and V floating-point numeric field type provides up to 20 significant digits.

Logical (L)

Paradox 5 and 7 and dBASE III+, IV, and V field type can store values representing True or False (yes or no). By default, valid entries include T and F (case is not important).

Memo (M)

Paradox 4, 5, and 7 as well as dBASE III+, IV, and V field. A Paradox field type is an Alpha variable-length field up to 256MB per field. dBASE Memo fields can contain binary as well as memo data.

OLE (O)

Paradox 1, 5, and 7 as well as dBASE V field type that can store OLE data.

Number (N)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V field type can store up to 15 significant digits -10307 to + 10308 with up to 15 significant digits.

dBASE number fields contain numeric data in a Binary Coded Decimal (BCD) format. Use number fields when you need to perform precise calculations on the field data. Calculations on number fields are performed more slowly but with greater precision than are calculations on float number fields. The size of a dBASE number field can be from 1 to 20. Remember, however, that BCD is in Paradox 5 and 7 only for compatibility and is mapped directly to the Number field type.

Short (S)

Paradox 3.5, 4, 5, and 7 field type that can contain integers from -- 32,767 through 32,767 (no decimal).

1.2.4 Paradox Standard Table Specifications

Also known as Paradox 4 table structure.

The Paradox standard table format was introduced in Paradox for DOS version 4. Other products that use the standard format include Paradox for DOS version 4.5, ObjectVision 2.1, and Paradox for Windows versions 1.0 and 4.5.

Earlier versions of the Paradox table type are referred to as the Compatible table type. In the BDE Configuration Utility, the level option for the Paradox driver dictates what default table type is created by Paradox for Windows. Use 3 for Compatible tables, 4 for Standard tables (the default). Following are the specifications for standard Paradox tables.

256MB file size limit if the table is in Paradox format and using a 4K block size.

Up to 255 fields per record.

Up to 64 validity checks per table.

A primary index can have up to 16 fields.

Tables can have up to 127 secondary indexes.

Up to two billion records per file. Because of the 256MB file size limit and other factors such as block size, however, the limit is much smaller. Tables of 190,000 records are easily achievable (and you can have more if you don't use up the 1,350-bytes-per-record limit for a keyed table). Tables with close to a million records are common.

Block size can be 1024, 2048, 3072, or 4096. Paradox stores data in fixed records. Even if part or all of the record is empty, the space is claimed. Knowing the interworkings can save you disk space. Paradox stores records in fixed blocks of 1024, 2048, 3072, 4096 in size.

After a block size is set for a table, that size is fixed, and all blocks in the table will be of that size. To conserve disk space, you want to try to get your record size as close to a multiple of block size as possible (minus 6 bytes, which are used by Paradox to manage the table).

Record size. 1,350 for keyed tables and 4,000 for unkeyed tables. When figuring out the size (the number of bytes or characters) of a table, remember that Alpha fields take up their size (for example, an A10 = 10 bytes), numeric field types take up 8 bytes, short number field types take up 2 bytes, money takes up 8, and dates take up 4 bytes.

Memos, BLOBs, and so on take 10 bytes plus however much of the memo is stored in the .DB. For example, M15 takes 25 bytes.

1.2.5 Paradox 5 Table Specifications

The Paradox 5 table format was introduced in Paradox for Windows version 5. Following are the specifications for Paradox 5 tables.

Up to two billion records per file.

File size is limited to two gigabytes.

Up to 255 fields per record.

Record size: Up to 10,800 bytes per record for indexed tables and 32,750 bytes per record for nonindexed tables. When figuring out the size (the number of bytes or characters) of a table, remember that Alpha fields take up their size (for example, an

A10 = 10 bytes), numeric field types take up 8 bytes, short number field types take up 2 bytes, money takes up 8, and dates take up 4 bytes.

Memos, BLOBs, and so on take 10 bytes plus however much of the memo is stored in the .DB. For example, M15 takes 25 bytes.

Up to 64 validity checks per table for Paradox for Windows tables.

A primary index can have up to 16 fields.

Tables can have up to 127 secondary indexes.

Block size can be from 1K to 32K in steps of 1K. For example, 1024, 2048, 3072, 4096, 5120...32768.

1.2.6 Paradox 7 and above Table Specifications

The Paradox 7 table format was introduced in Paradox version 7 for Windows 95/NT. The Paradox 7 table format has all the same specifications as the Paradox 5 table format with two additions. Following are the specification additions for the Paradox 7 table format.

- Added descending secondary indexes.
- Added unique secondary indexes

1.2.7 Paradox Field Types

Alpha (A)

Paradox 3.5, 4, 5, and 7 field type that can contain up to 255 letters and numbers. This field type was called Alphanumeric in versions of Paradox before version 5. It is similar to the Character field type in dBASE.

Autoincrement (+)

Field type introduced in the Paradox 5 table format that adds one to the highest number in the table whenever a record is inserted. The starting range can from -2,147,483,647 to 2,147,483,647. Deleting a record does not change the field values of other records.

BCD (#)

Paradox 5 and 7 field type which is provided only for compatibility with other applications that use BCD data. Paradox correctly interprets BCD data from other applications that use the BCD type. When Paradox performs calculations on BCD data, it converts the data to the numeric float type, then converts the result back to BCD. When this field type is fully supported, it will support up to 32 significant digits.

Binary (B)

Paradox 1, 5, and 7 field type that can store binary data up to 256MB per field.

Bytes (Y)

Paradox 5 and 7 field type for storing binary data up to 255 bytes. Unlike binary fields, bytes fields are stored in the Paradox table (rather than in the separate .MB file), allowing for faster access.

Date (D)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V. dBASE tables can store dates from January 1, 100, to December 31, 9999. Paradox 5 tables can store from 12/31/9999 B.C. to 12/31/9999 A.D.

Formatted Memo (F)

Paradox 1, 4.5, 5, and 7 field type is like a memo field except that you can format the text. You can alter and store the text attributes of typeface, style, color, and size. This rich text document has a variable-length up to 256MB per field.

Graphic (G)

Paradox 1, 5, and 7 field type can contain pictures in .BMP (up to 24 bit), .TIF (up to 256 color), .GIF (up to 256 color), .PCX, and .EPS file formats. Not all graphic variations are available. For example, currently you cannot store a 24-bit .TIF graphic. When you paste a graphic into a graphic field, Paradox converts the graphic into the .BMP format.

Logical (L)

Paradox 5 and 7 and dBASE III+, IV, and V field type can store values representing True or False (yes or no). By default, valid entries include T and F (case is not important).

Memo (M)

Paradox 4, 5, and 7 as well as dBASE III+, IV, and V field. A Paradox field type is an Alpha variable-length field up to 256MB per field. dBASE Memo fields can contain binary as well as memo data.

For Paradox tables, the file is divided into blocks of 512 characters. Each block is referenced by a sequential number, beginning at zero. Block 0 begins with a 4-byte number in hexadecimal format, in which the least significant byte comes first. This number specifies the number of the next available block. It is, in effect, a pointer to the end of the memo file. The remainder of Block 0 isn't used.

Money ($)

Paradox 3.5, 4, 5, and 7 field type, like number fields, can contain only numbers. They can hold positive or negative values. Paradox recognizes up to six decimal places when performing internal calculations on money fields. This field type was called Currency in previous versions of Paradox.

OLE (O)

Paradox 1, 5, and 7 as well as dBASE V field type that can store OLE data.

Number (N)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V field type can store up to 15 significant digits -10307 to + 10308 with up to 15 significant digits.

dBASE number fields contain numeric data in a Binary Coded Decimal (BCD) format. Use number fields when you need to perform precise calculations on the field data. Calculations on number fields are performed more slowly but with greater precision than are calculations on float number fields. The size of a dBASE number field can be from 1 to 20. Remember, however, that BCD is in Paradox 5 and 7 only for compatibility and is mapped directly to the Number field type.

Short (S)

Paradox 3.5, 4, 5, and 7 field type that can contain integers from -- 32,767 through 32,767 (no decimal).

Time (T)

Paradox 5 and 7 field type that can contain time times of day, stored in milliseconds since midnight and limited to 24 hours.

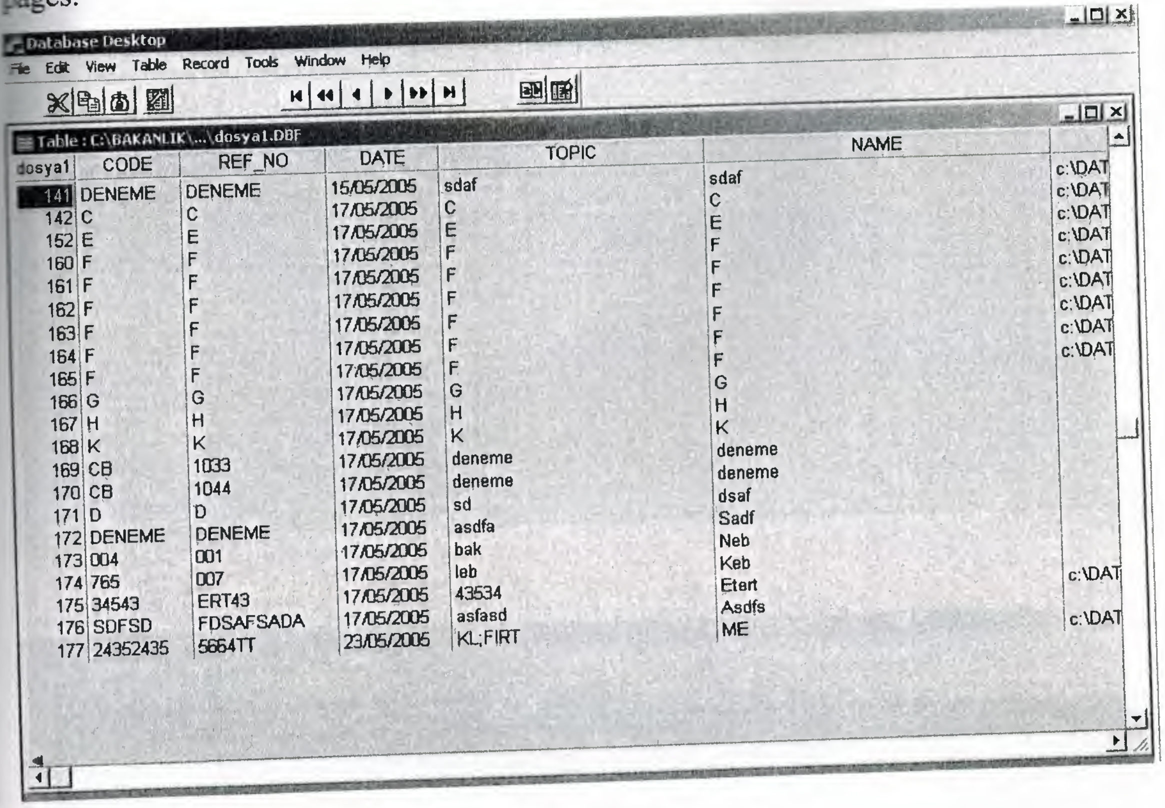
This field type does not store duration which is the difference between two times. For example, if you need to store the duration of a song, use an Alpha field. Whenever you need to store time, make a distinction between clock time and duration. The Time field type is perfect for clock time. Duration can be stored in an Alpha field and manipulated with code.

TimeStamp (@)

Paradox 5 field type comprised of both date and time values. Rules for this field type are the same as those for date fields and time fields.

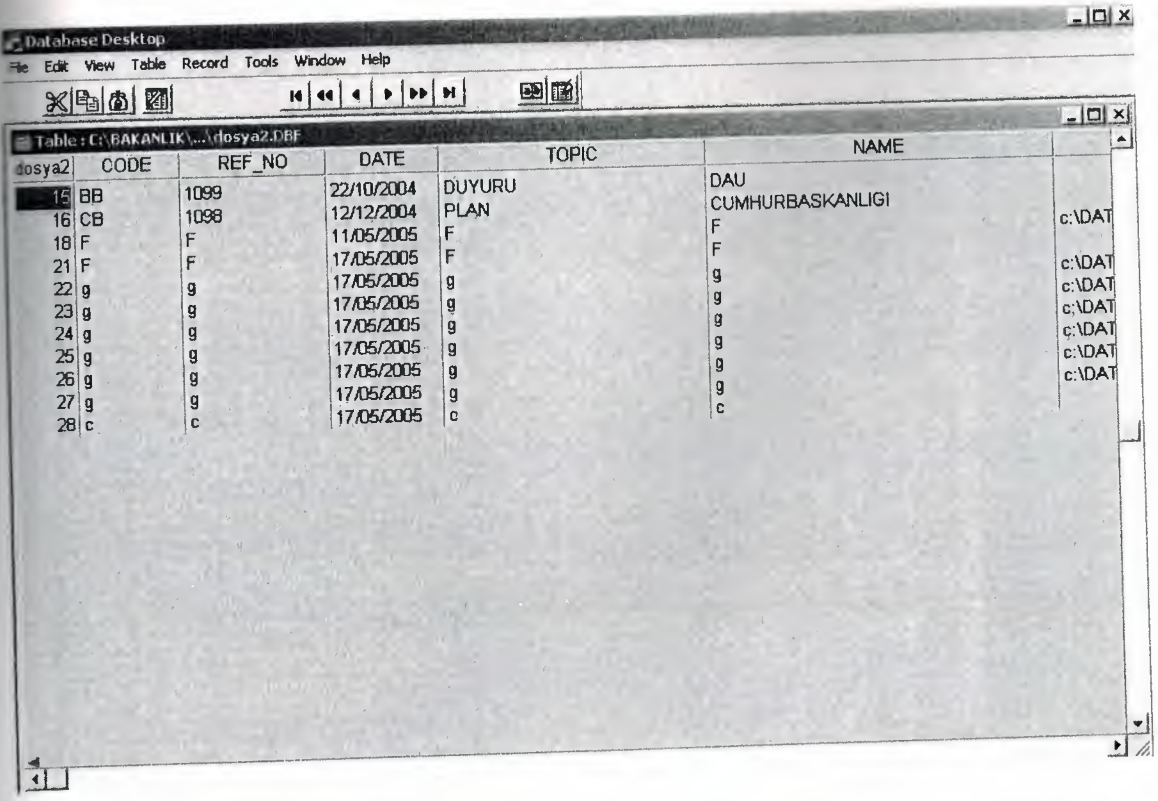
1.2.8 About Our Database

In our archive program we used dbase for database .You can see the tables in following pages.

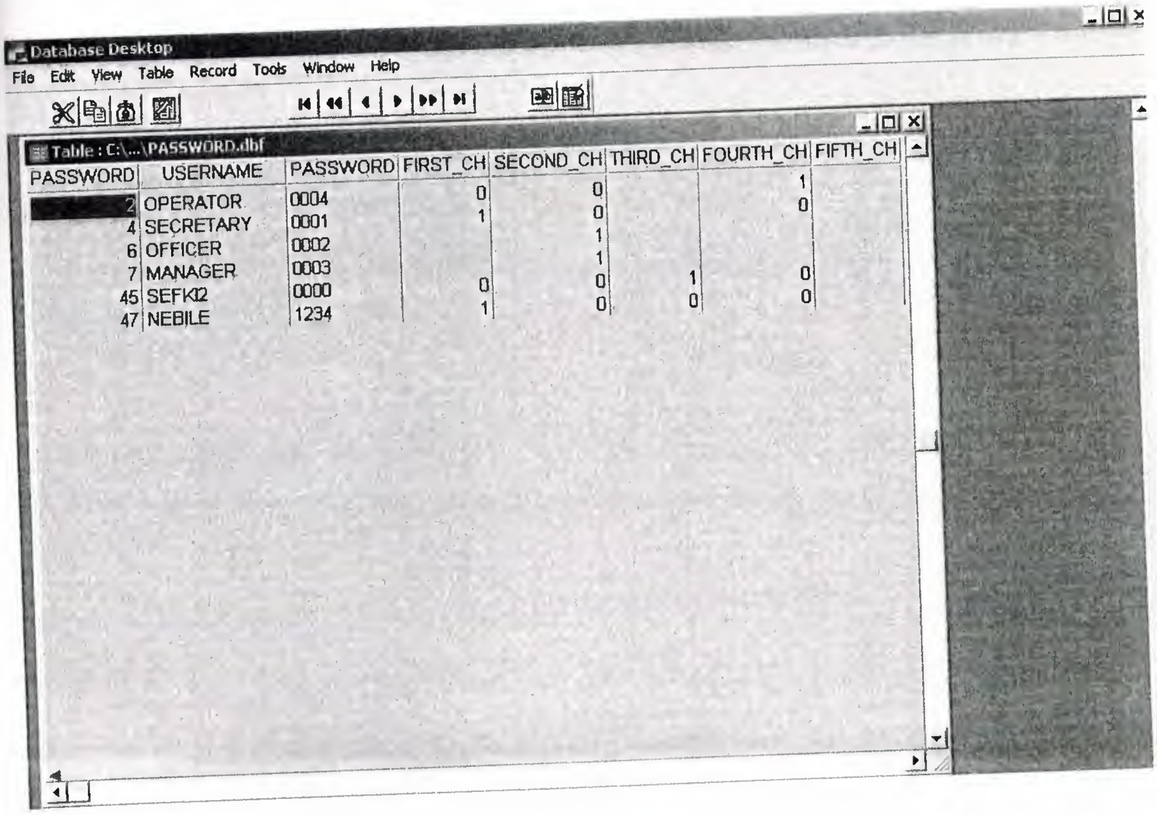


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\BAKANLIK\...\dosya1.DBF

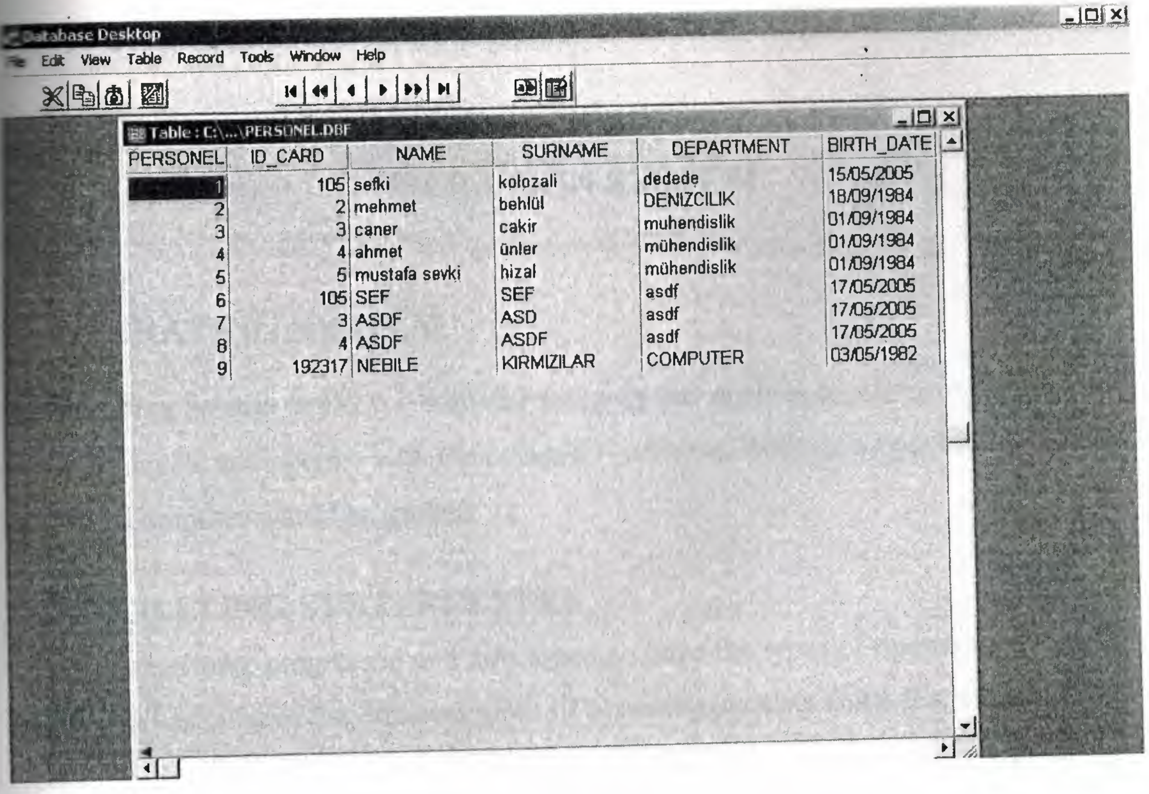
dosya1	CODE	REF_NO	DATE	TOPIC	NAME	
141	DENEME	DENEME	15/05/2005	sdaf	sdaf	c:\DAT
142	C	C	17/05/2005	C	C	c:\DAT
152	E	E	17/05/2005	E	E	c:\DAT
160	F	F	17/05/2005	F	F	c:\DAT
161	F	F	17/05/2005	F	F	c:\DAT
162	F	F	17/05/2005	F	F	c:\DAT
163	F	F	17/05/2005	F	F	c:\DAT
164	F	F	17/05/2005	F	F	c:\DAT
165	F	F	17/05/2005	F	F	
166	G	G	17/05/2005	G	G	
167	H	H	17/05/2005	H	H	
168	K	K	17/05/2005	K	K	
169	CB	1033	17/05/2005	deneme	deneme	
170	CB	1044	17/05/2005	deneme	deneme	
171	D	D	17/05/2005	sd	dsaf	
172	DENEME	DENEME	17/05/2005	asdfa	Sadf	
173	004	001	17/05/2005	bak	Neb	
174	765	007	17/05/2005	leb	Keb	c:\DAT
175	34543	ERT43	17/05/2005	43534	Etert	
176	SDFSD	FDSAFSADA	17/05/2005	asfasd	Asdfs	c:\DAT
177	24352435	5664TT	23/05/2005	KL;FIRT	ME	


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\BAKANLIK\...\dosya2.DBF

dosya2	CODE	REF_NO	DATE	TOPIC	NAME	
15	BB	1099	22/10/2004	DUYURU	DAU	
16	CB	1098	12/12/2004	PLAN	CUMHURBASKANLIGI	
18	F	F	11/05/2005	F	F	c:\DAT
21	F	F	17/05/2005	F	F	
22	g	g	17/05/2005	g	g	c:\DAT
23	g	g	17/05/2005	g	g	c;\DAT
24	g	g	17/05/2005	g	g	c:\DAT
25	g	g	17/05/2005	g	g	c:\DAT
26	g	g	17/05/2005	g	g	c:\DAT
27	g	g	17/05/2005	g	g	c:\DAT
28	c	c	17/05/2005	c	c	


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\...\PASSWORD.dbf

PASSWORD	USERNAME	PASSWORD	FIRST_CH	SECOND_CH	THIRD_CH	FOURTH_CH	FIFTH_CH
2	OPERATOR	0004	0	0		1	
4	SECRETARY	0001	1	0		0	
6	OFFICER	0002		1			
7	MANAGER	0003		1			
45	SEFKI2	0000	0	0	1	0	
47	NEBILE	1234	1	0	0	0	


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\...\PERSONEL.DBF

PERSONEL	ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
1	105	sefki	kolozali	dedede	15/05/2005
2	2	mehmet	behlül	DENIZCILIK	18/09/1984
3	3	caner	cakir	muhendislik	01/09/1984
4	4	ahmet	ünler	mühendislik	01/09/1984
5	5	mustafa sevki	hizal	mühendislik	01/09/1984
6	105	SEF	SEF	asdf	17/05/2005
7	3	ASDF	ASD	asdf	17/05/2005
8	4	ASDF	ASDF	asdf	17/05/2005
9	192317	NEBILE	KIRMIZILAR	COMPUTER	03/05/1982

CHAPTER 2

OPERATION SYSTEM

2.1 OPERATING SYSTEM

An Operating System or OS is a software program that enables the computer hardware to communicate and operate with the computer software. Without a computer Operating System a computer would be useless.

2.2 OPERATING SYSTEM TYPES

As computers have progressed and developed so have the types of operating systems. Below is a basic list of the different types of operating systems and a few examples of Operating Systems that fall into each of the categories. Many computer Operating Systems will fall into more then one of the below categories.

2.2.1 Multi-user

A multi-user Operating System allows for multiple users to use the same computer at the same time and/or different times.

Below are some examples of multi-user Operating Systems.

Linux
UNIX
Windows 2000

2.2.2 Multiprocessing

An Operating System capable of supporting and utilizing more than one computer processor.

1. When referring to a network, a multi-user system is a term commonly used to define a computer capable of allowing multiple users to connect to a network.

2. When referring to a computer Operating System, a multi-user system is a computer with an Operating system that supports multiple users at once and/or different times.

Below are some examples of multiprocessing Operating Systems.

Linux
UNIX
Windows 2000

2.2.3 Multitasking

An Operating systems that is capable of allowing multiple software processes to be run at the same time. Below are some examples of multitasking Operating Systems.

UNIX
Windows 2000

2.2.4 Multithreading

Operating systems that allow different parts of a software program to run concurrently. Operating systems that would fall into this category are:

Linux
UNIX
Windows 2000

2.3 Introduction to UNIX:

2.3.1 Objectives

Its covers:

- The concept of an operating system.
- The internal architecture of an operating system.
- The evolution of the UNIX operating system into two broad schools (BSD and SYSV) and the development of Linux, a popular open source operating system.

- The architecture of the Linux operating system in more detail.
- How to log into (and out of) UNIX and change your password.
- The general format of UNIX commands.

2.3.2 What is an Operating System?

An operating system (OS) is a resource manager. It takes the form of a set of software routines that allow users and application programs to access system resources (e.g. the CPU, memory, disks, modems, printers network cards etc.) in a safe, efficient and abstract way.

For example, an OS ensures safe access to a printer by allowing only one application program to send data directly to the printer at any one time. An OS encourages efficient use of the CPU by suspending programs that are waiting for I/O operations to complete to make way for programs that can use the CPU more productively. An OS also provides convenient abstractions (such as files rather than disk locations) which isolate application programmers and users from the details of the underlying hardware.

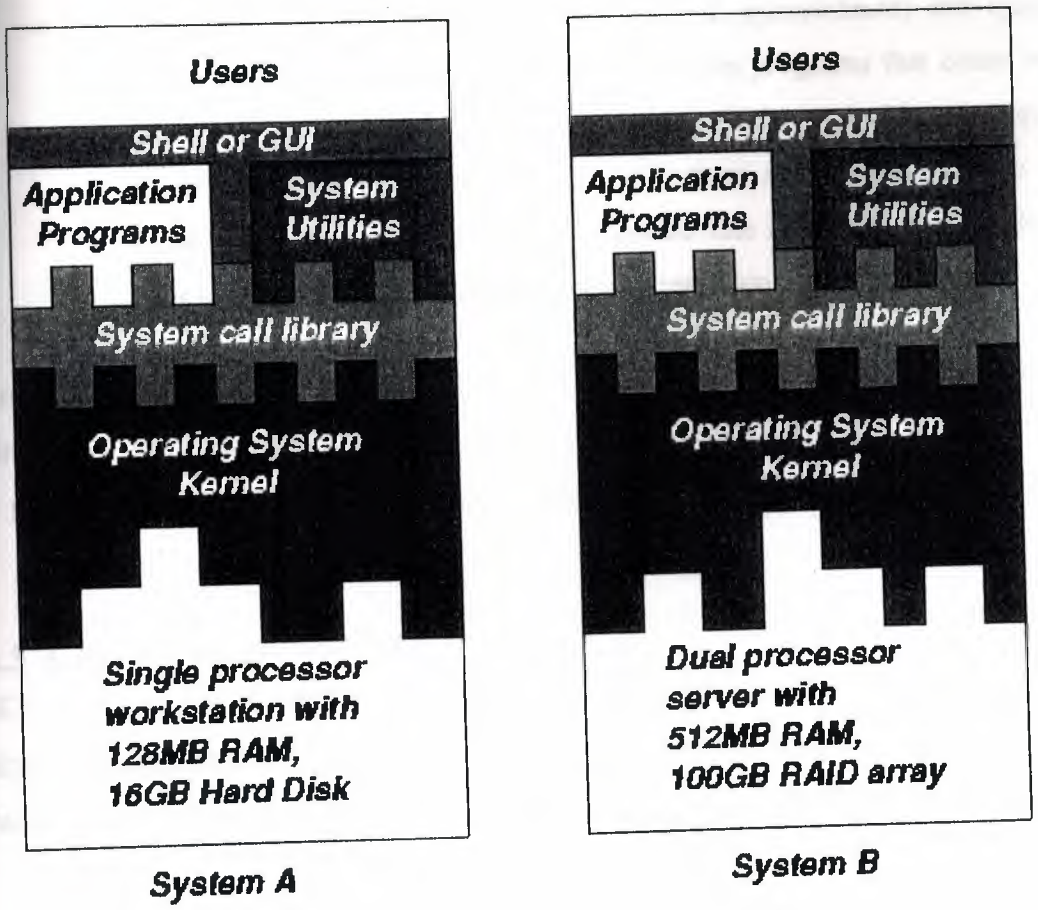


Fig. 1.1: General operating system architecture

Fig. 1.1 presents the architecture of a typical operating system and shows how an OS succeeds in presenting users and application programs with a uniform interface without regard to the details of the underlying hardware. We see that:

- The operating system kernel is in direct control of the underlying hardware. The kernel provides low-level device, memory and processor management functions (e.g. dealing with interrupts from hardware devices, sharing the processor among multiple programs, allocating memory for programs etc.)
- Basic hardware-independent kernel services are exposed to higher-level programs through a library of system calls (e.g. services to create a file, begin execution of a program, or open a logical network connection to another computer).

- Application programs (e.g. word processors, spreadsheets) and system utility programs (simple but useful application programs that come with the operating system, e.g. programs which find text inside a group of files) make use of system calls. Applications and system utilities are launched using a shell (a textual command line interface) or a graphical user interface that provides direct user interaction.

Operating systems (and different flavours of the same operating system) can be distinguished from one another by the system calls, system utilities and user interface they provide, as well as by the resource scheduling policies implemented by the kernel.

2.3.3 A Brief History of UNIX

UNIX has been a popular OS for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities. What follows here is a simplified history of how UNIX has developed.

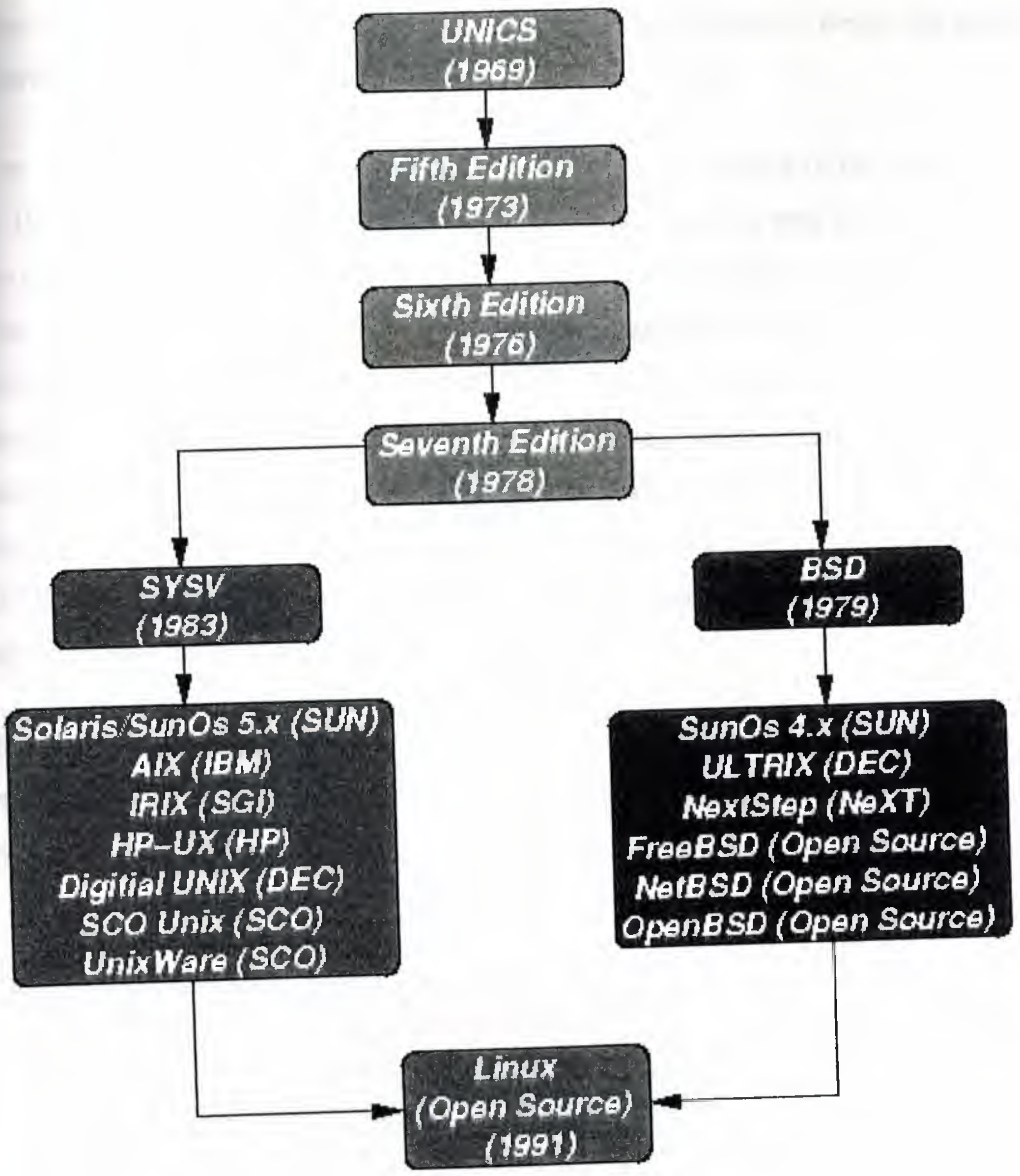


Fig. 1.2: Simplified UNIX FamilyTree

In the late 1960s, researchers from General Electric, MIT and Bell Labs launched a joint project to develop an ambitious multi-user, multi-tasking OS for mainframe computers known as MULTICS (Multiplexed Information and Computing System). MULTICS failed (for some MULTICS enthusiasts "failed" is perhaps too strong a word to use here), but it did inspire Ken Thompson, who was a researcher at Bell Labs, to have a go at writing a simpler operating system himself. He wrote a simpler version of MULTICS on a PDP7 in assembler and called his attempt UNICS (Uniplexed Information and Computing System). Because memory and CPU power were at a premium in those days, UNICS (eventually shortened to UNIX) used short commands to minimize the

space needed to store them and the time needed to decode them - hence the tradition of short UNIX commands we use today, e.g. ls, cp, rm, mv etc.

Ken Thompson then teamed up with Dennis Ritchie, the author of the first C compiler in 1973. They rewrote the UNIX kernel in C - this was a big step forwards in terms of the system's portability - and released the Fifth Edition of UNIX to universities in 1974. The Seventh Edition, released in 1978, marked a split in UNIX development into two main branches: SYSV (System 5) and BSD (Berkeley Software Distribution). BSD arose from the University of California at Berkeley where Ken Thompson spent a sabbatical year. Its development was continued by students at Berkeley and other research institutions. SYSV was developed by AT&T and other commercial companies. UNIX flavours based on SYSV have traditionally been more conservative, but better supported than BSD-based flavours.

The latest incarnations of SYSV (SVR4 or System 5 Release 4) and BSD Unix are actually very similar. Some minor differences are to be found in file system structure, system utility names and options and system call libraries as shown in Fig 1.3.

Feature	**Typical SYSV**	**Typical BSD**
kernel name	/unix	/vmunix
boot init	/etc/rc.d directories	/etc/rc.* files
mounted FS	/etc/mnttab	/etc/mtab
default shell	sh, ksh	csh, tcsh
FS block size	512 bytes->2K	4K->8K
print subsystem	lp, lpstat, cancel	lpr, lpq, lprm
echo command (no new line)	echo "\c"	echo -n
ps command	ps -fae	ps -aux
multiple wait syscalls	poll	select
memory access syscalls	memset, memcpy	bzero, bcopy

Fig. 1.3: Differences between SYSV and BSD

Linux is a free open source UNIX OS for PCs that was originally developed in 1991 by Linus Torvalds, a Finnish undergraduate student. Linux is neither pure SYSV or pure BSD. Instead, incorporates some features from each (e.g. SYSV-style startup files but BSD-style file system layout) and aims to conform with a set of IEEE standards called POSIX (Portable Operating System Interface). To maximise code portability, it typically supports SYSV, BSD and POSIX system calls (e.g. poll, select, memset, memcpy, bzero and bcopy are all supported).

The open source nature of Linux means that the source code for the Linux kernel is freely available so that anyone can add features and correct deficiencies. This approach has been very successful and what started as one person's project has now turned into a collaboration of hundreds of volunteer developers from around the globe. The open source approach has not just successfully been applied to kernel code, but also to application programs for Linux.

As Linux has become more popular, several different development streams or distributions have emerged, e.g. Redhat, Slackware, Mandrake, Debian, and Caldera. A distribution comprises a prepackaged kernel, system utilities, GUI interfaces and application programs.

Redhat is the most popular distribution because it has been ported to a large number of hardware platforms (including Intel, Alpha, and SPARC), it is easy to use and install and it comes with a comprehensive set of utilities and applications including the X Windows graphics system, GNOME and KDE GUI environments, and the StarOffice suite (an open source MS-Office clone for Linux).

2.3.4 Architecture of the Linux Operating System

Linux has all of the components of a typical OS (at this point you might like to refer back to Fig 1.1):

- **Kernel**

  The Linux kernel includes device driver support for a large number of PC hardware devices (graphics cards, network cards, hard disks etc.), advanced processor and memory management features, and support for many different types of filesystems (including DOS floppies and the

ISO9660 standard for CDROMs). In terms of the services that it provides to application programs and system utilities, the kernel implements most BSD and SYSV system calls, as well as the system calls described in the POSIX.1 specification.

The kernel (in raw binary form that is loaded directly into memory at system startup time) is typically found in the file /boot/vmlinuz, while the source files can usually be found in /usr/src/linux.The latest version of the Linux kernel sources can be downloaded from http://www.kernel.org.

- **Shells and GUIs**
  Linux supports two forms of command input: through textual command line shells similar to those found on most UNIX systems (e.g. sh - the Bourne shell, bash - the Bourne again shell and csh - the C shell) and through graphical interfaces (GUIs) such as the KDE and GNOME window managers. If you are connecting remotely to a server your access will typically be through a command line shell.

- **System Utilities**
  Virtually every system utility that you would expect to find on standard implementations of UNIX (including every system utility described in the POSIX.2 specification) has been ported to Linux. This includes commands such as ls, cp, grep, awk, sed, bc, wc, more, and so on. These system utilities are designed to be powerful tools that do a single task extremely well (e.g. grep finds text inside files while wc counts the number of words, lines and bytes inside a file). Users can often solve problems by interconnecting these tools instead of writing a large monolithic application program.

Like other UNIX flavours, Linux's system utilities also include server programs called daemons which provide remote network and administration services (e.g. telnetd and sshd provide remote login facilities, lpd provides printing services, httpd serves web

pages, crond runs regular system administration tasks automatically). A daemon (probably derived from the Latin word which refers to a beneficient spirit who watches over someone, or perhaps short for "Disk And Execution MONitor") is usually spawned automatically at system startup and spends most of its time lying dormant (lurking?) waiting for some event to occur.

- **Application programs**

  Linux distributions typically come with several useful application programs as standard. Examples include the emacs editor, xv (an image viewer), gcc (a C compiler), g++ (a C++ compiler), xfig (a drawing package), latex (a powerful typesetting language) and soffice (StarOffice, which is an MS-Office style clone that can read and write Word, Excel and PowerPoint files).

  Redhat Linux also comes with rpm, the Redhat Package Manager which makes it easy to install and uninstall application programs.

2.4 An Introduction to Linux

2.4.1 What is Linux?

Linux is a UNIX-based operating system originally developed as for Intel-compatible PC's. It is now available for most types of hardware platforms, ranging from PDAs (and according to some reports, a wristwatch) to mainframes. Linux is a "modern operating system", meaning it has such features as virtual memory, memory protection, and preemptive multitasking.

Linux is built and supported by a large international community of developers and users dedicated to free, open-source software. This community sees Linux as an alternative to such proprietary systems as Windows and Solaris, and as a platform for alternatives to such proprietary applications as MS Office, Internet Explorer, and Outlook.

As a result of this community, there is a very large collection of free software available for Linux. There are graphical environments (GUIs), office applications, developers'

tools, system utilities, business applications, document publishing tools, network client and server applications -- the list goes on.

The best part of this community is that all code is open. This means there is no barrier to entry; for any given problem, there are generally several applications that solve the problem. These applications can also borrow the best parts from each other to become even better. An excellent example of this is Galeon. Galeon is a web browser which took Mozilla's web page rendering engine and integrated it with a GTK frontend (instead of Mozilla's normal frontend).

Linux specifically refers to the Linux kernel. However, the kernel is useless without a set of tools and applications to run on the kernel. Linux is most commonly distributed with this toolset and a collection of applications in what is called a "distribution". The most common are Redhat, Mandrake, Suse, and Debian. Distributions differ in three basic ways: the process for installing the distribution, the applications available, and process for installing and managing these applications.

2.4.2 Why use Linux?

Reasons to Install Linux

- Configurability
- Convenience
- Stability
- Community
- Freedom

2.4.2.1 Configurability

Linux distributions give the user full access to configure just about any aspect of their system. Options range from the simple and straightforward (for instance, changing the background image) to the more esoteric (for instance, making the "Caps Lock" key behave like "Control"). Almost any aspect of the user experience can be configured.

Linux also allows the user to automate just about any task. Advanced scripting and high-level programming are standard features. Most operations are accessible via these scripting options. Finally, Linux offers the ultimate in configurability: the source code, to be modified as you see fit.

2.4.2.2 Convenience

While Linux takes some effort to get set up, once it is set up, it is surprisingly low-maintenance. Package management can simply be a matter of running two commands in the shell. Linux also offers complete remote access. This allows the user to act exactly as if she is sitting at that computer's desk, potentially across town or on the other side of the world.

2.4.2.3 Stability

Linux is based on the UNIX kernel. It provides preemptive multitasking and protected memory. Preemptive multitasking prevents any application from permanently stealing the CPU and locking up the machine. Protected memory prevents applications from interfering with and crashing one-another.

Linux and related tools are also open-source. This means that the source code is available for the public to view. There are literally hundreds, if not thousands, of developers working on the various pieces of Linux. In this open development process, bugs are fixed very quickly. In addition, bugs are fixed immediately, instead of waiting for the next major release. It certainly helps that the people who develop Linux and associated tools use their programs every day.

2.4.2.4 Community

Linux is part of the greater open-source community. This consists of thousands of developers and many more users world-wide who support open software. This user and developer base is also a support base.

In Rice, there is the Rice Linux Users Group (the group who are bringing you this class). We hold regular meetings where people can bring up their Linux problems. There is also the newsgroup rice.comp.linux, where questions can be asked or problems laid out any time, day or night. This newsgroup is mirrored to the RLUG-discuss mailing list, for RLUG members who don't have access to Rice newsgroups.

Worldwide, the Linux community is even greater. There is a mailing list for just about every project or piece of software in active development -- if you have a question about

a program, who better to ask than the person who wrote it? There are also newsgroups and web pages which have collectively with the mailing lists probably addressed every problem someone new to Linux has encountered several times over.

2.4.2.5 Freedom

Linux is free. This means more than just costing nothing. This means that you are allowed to do whatever you want to with the software. This is why Redhat, Mandrake, and Suse are all allowed to sell their own distributions of Linux. The only restriction placed on Linux is that, if you distribute Linux, you must grant all the privileges to the code that you had, including providing the source. This prevents a corporation from using the Linux kernel as the basis for their proprietary operating system.

2.5 The differences between Linux and other UNIX

The differences between Linux and other UNIX operating systems is barely noticable to the average user. Commands might be located in slightly different places or may have different sets of arguments but the functionallity is the same. With the GNU tools available for most platforms these days even these differences are starting to vanish. Most of the differences are located in the kernel design and the systems administration tools. Some systems such as AIX and HP-UX provide fancy graphical management programs while others such as Linux and Solaris tend to rely on editing files and running commands manually. To sum up, UNIX is like a car: once you can drive one, you can drive them all. However if you open up the bonnet to tinker around you'll find many diverse systems as you move from one type to another.

CHAPTER 3

DELPHI

3.1.INTRODUCTION

The name "Delphi" was never a term with which either Olaf Helmer or Norman Dalkey (the founders of the method) were particular happy. Since many of the early Delphi studies focused on utilizing the technique to make forecasts of future occurrences, the name was first applied by some others at Rand as a joke. However, the name stuck. The resulting image of a priestess, sitting on a stool over a crack in the earth, inhaling sulfur fumes, and making vague and jumbled statements that could be interpreted in many different ways, did not exactly inspire confidence in the method.

The straightforward nature of utilizing an iterative survey to gather information "sounds" so easy to do that many people have done "one" Delphi, but never a second. Since the name gives no obvious insight into the method and since the number of unsuccessful Delphi studies probably exceeds the successful ones, there has been a long history of diverse definitions and opinions about the method. Some of these misconceptions are expressed in statements such as the following that one finds in the literature:

It is a method for predicting future events.

It is a method for generating a quick consensus by a group.

It is the use of a survey to collect information.

It is the use of anonymity on the part of the participants.

It is the use of voting to reduce the need for long discussions.

It is a method for quantifying human judgement in a group setting.

Some of these statements are sometimes true; a few (e.g. consensus) are actually contrary to the purpose of a Delphi. Delphi is a communication structure aimed at

producing detailed critical examination and discussion, not at forcing a quick compromise. Certainly quantification is a property, but only to serve the goal of quickly identifying agreement and disagreement in order to focus attention. It is often very common, even today, for people to come to a view of the Delphi method that reflects a particular application with which they are familiar. In 1975 Linstone and Turoff proposed a view of the Delphi method that they felt best summarized both the technique and its objective:

"Delphi may be characterized as a method for structuring a group communication process, so that the process is effective in allowing a group of individuals, as a whole, to deal with complex problems." (page 3)

The essence of Delphi is structuring of the group communication process. Given that there had been much earlier work on how to facilitate and structure face-to-face meetings, the other important distinction was that Delphi was commonly applied utilizing a paper and pencil communication process among groups in which the members were dispersed in space and time. Also, Delphis were commonly applied to groups of a size (30 to 100 individuals) that could not function well in a face-to-face environment, even if they could find a time when they all could get together.

Additional opportunity has been added by the introduction of Computer Mediated Communication Systems (Hiltz and Turoff, 1978; Rice and Associates, 1984; Turoff, 1989; Turoff, 1991). These are computer systems that support group communications in either a synchronous (Group Decision Support Systems, Desanctis et. al., 1987) or an asynchronous manner (Computer Conferencing). Techniques that were developed and refined in the evolution of the Delphi Method (e.g. anonymity, voting) have been incorporated as basic facilities or tools in many of these computer based systems. As a result, any of these systems can be used to carry out some form of a Delphi process or Nominal Group Technique (Delbecq, et. al., 1975).

The result, however, is not merely confusion due to different names to describe the same things; but a basic lack of knowledge by many people working in these areas as to what was learned in the studies of the Delphi Method about how to properly employ these techniques and their impact on the communication process. There seems to be a great deal of "rediscovery" and repeating of earlier misconceptions and difficulties.

Given this situation, the primary objective of this chapter is to review the specific properties and methods employed in the design and execution of Delphi Exercises and to examine how they may best be translated into a computer based environment.

3.2. ASYNCHRONOUS INTERACTION

Perhaps the most important and least understood property of the Delphi method is the ability of members of a group to participate in an asynchronous manner. This property of asynchronous interaction has two characteristics:

A person may choose to participate in the group communication process when they feel they want to.

A person may choose to contribute to that aspect of the problem to which they feel best able to contribute.

It does not matter what time of the day or night Delphi participants think of good ideas to include in their response. They can fill out a Delphi survey when they wish to, or they can go to a computer terminal to contribute when they wish to. This can be done at whatever point in time the individual feels he or she has thought of significant things to include in response to the issues involved. Participants can revise and add to their responses over time, before sending them to the group monitor for dissemination to the others.

A good Delphi survey attempts to tackle the problem from many different perspectives. Sometimes this is referred to as including questions in the Delphi survey which approach the problem both from the "bottom-up" and from the "top-down" perspectives. This allows different individuals in the group to focus on the approach to problem solving with which they feel most comfortable.

In a normal face-to-face group process, and in the environment characterized by face-to-face Group Decision Support Systems, all the members of the group are forced into a lockstep treatment of a problem. When the group is considering the subject of "goals," those who have difficulty dealing with "abstraction" may feel at a disadvantage, because they do not have as much to contribute. Conversely, when focusing on specific solution approaches, those who deal better with "abstraction" may not feel they are contributing. One of the specific advantages of groups is to allow individuals with differing perspectives and/or differing cognitive abilities to contribute to those parts of a complex

problem for which they have both the appropriate knowledge and appropriate problem solving skills. A typical model for a group problem solving process is:

Recognition of the problem
Defining the problem
Changing the representation of the problem
Developing the goals associated with solving the problem
Determining the strategy for generating the possible solutions
Choosing a strategy
Generating the evaluation criteria to be applied to solutions
Evaluating the solution criteria
Generating the solutions
Evaluating the solutions

The literature on cognitive abilities and human problem solving does confirm that individuals differ considerably, based upon their cognitive abilities (Benbasat and Taylor, 1982; Streitz, 1987), in their ability to deal with different aspects of a problem solving situation. This depends upon such psychological dimensions as their ability to deal with Abstraction - No Abstraction, Search - No Search, Data Driven - Conceptually Driven, and Deductive - Inductive cognitive processes.

In most face-to-face approaches, the group is forced as a whole to take a sequential path through a group problem solving process. In the Delphi process, we try to design a communication structure that allows any individual to choose the sequence in which to examine and contribute to the problem solving process. This is the single most important criterion by which we should evaluate the design of a Delphi oriented communication structure. Does it allow the individual to exercise personal judgement about what part of the problem to deal with at any time in the group problem solving process?

It is actually easier to accomplish this using a computer system than it has been with paper and pencil based Delphi studies. The "round" structure and the need to limit the physical size of any paper and pencil survey places severe constraints on the degree to which one can carry out the above approach. Hence, paper and pencil Delphis are usually limited by the "top-down/bottom-up" dichotomy rather than allowing more complete parallel entry to any aspect of the problem. For example, in a single Delphi one might explore on the first round "goals" (a top view) and specific "consequences" (a

bottom view). Relating goals to consequences requires developing the relationships inherent in alternative actions and states of nature. These would be put off to a later round. In the computerized environment individuals could be free to tackle any aspect of the problem according to personal preferences.

This particular objective of Delphi design is also characterized by two other practices commonly applied to Delphi studies. First, it should be clear to the respondents that they do not have to respond to every question, but can decide to take a "no judgement" view. Secondly, one usually solicits the respondent's confidence in their judgements, particularly when they are quantified judgements. This has been found to improve the quality of the estimates made in Delphi exercises (Dalkey, 1970). This allows the respondents to estimate their own degree of expertise on the judgements they are supplying. The fact that contributions can be made anonymously also means a person does not have to feel embarrassment if he or she does not feel able to confidently contribute to a specific aspect of the problem.

This advantage for the Delphi approach comes at an obvious price. With material being supplied in parallel, it is clear that the need to structure and organize it in a manner that it makes sense to the group is a primary requirement (Turoff, 1974, 1991; Hiltz and Turoff, 1985). The need to carefully define the total communication structure and put it into a framework that produces both a group view and a synchronization of the group process is the most difficult part of a good Delphi design. We will treat this in following sections. In paper and pencil Delphis, this is the effort that must be undertaken by the design team in processing the results of each round and producing a proper summary. In a computer based Delphi process, this has a somewhat different connotation in that the round structure disappears, replaced by a continuous feedback process which may or may not involve human intervention for the processing.

The most significant observation resulting from the above considerations, is that most of the attempts to understand the group problem solving process in the computer based environment are still based upon models that were developed from studying face-to-face groups. Thus, what are often thought of as being "ideal" group problem solving structures are based upon the "sequential" treatment of a problem by a group (Turoff, 1991). There has been little work to date to develop models of the group problem solving process that are based upon parallel and asynchronous activities by the individuals within the group. There is need for a model which integrates the individual

problem solving process with the group process. It is only within the context of such a model that we can come to a deeper understanding of the design process that goes beyond the trial and error evolution of the method that has occurred to date.

3.3 ANONYMITY

Perhaps the property that most characterizes the Delphi method in the mind of most people is the use of anonymity. Typically, in paper and pencil Delphis there is no identification of who contributed specific material or who made a particular evaluative judgment about it. This property is not one that should be considered a hard and fast rule for all aspects of a Delphi. Moreover, the computer makes possible variations in anonymity not possible in a paper and pencil environment. Before we explore these, we should look at the primary reasons for anonymity:

Individuals should not have to commit themselves to initial expressions of an idea that may not turn out to be suitable.

If an idea turns out to be unsuitable, no one loses face from having been the individual to introduce it.

Persons of high status are reluctant to produce questionable ideas.

Committing one's name to a concept makes it harder to reject it or change one's mind about it.

Votes are more frequently changed when the identity of a given voter is not available to the group.

The consideration of an idea or concept may be biased by who introduced it.

When ideas are introduced within a group where severe conflicts exist in either "interests" or "values," the consideration of an idea may be biased by knowing it is produced by someone with whom the individual agrees or disagrees.

The high social status of an individual contributor may influence others in the group to accept the given concept or idea.

Conversely, lower status individuals may not introduce ideas, for fear that the idea will be rejected outright.

In essence, the objective of anonymity is to allow the introduction and evaluation of ideas and concepts by removing some of the common biases normally occurring in the face-to-face group process. Sometimes the use of anonymity has been carried too far.

For example, it is important that the members of a Delphi exercise believe that they are communicating with a peer group. An individual participant must feel that the other members of the group will be able to contribute valuable insight about the problem being examined. This is a primary factor in motivating participation. It is usual to inform the participants about who is actually involved in the group of Delphi respondents. Only when there are strong antagonisms among group members would one consider not doing this.

Delphi panelists are motivated to participate actively only if they feel they will obtain value from the information they receive as a result of the process. This value received needs to be at least equal, in their minds, to the effort expended to contribute information. This is one reason why blanket invitations to participate in a Delphi that do not specify who will be involved and what the feedback will be to the group members often result in very low participation rates.

When one introduces the concept of conducting a Delphi through a Computer Mediated Communication System, there are more options available for handling the process of anonymity. First, one can easily incorporate the use of pen-names (Hiltz, Turoff, and Johnson, 1989). While this does not identify who a person is, it does allow a person to be identified with a set of related contributions. This allows the other members of a group to obtain more understanding of why specific individuals are agreeing or disagreeing with certain concepts. For example, knowing all the arguments a person has made about accepting or rejecting a given position allows people to better tailor what needs to be said to perhaps change an individual's viewpoint. It also allows the expression of more complex individual viewpoints. This coherency is hard to observe or utilize when everything is anonymous.

As a result, it is probably desirable in most computer based Delphis to impose the default use of pen-names rather than anonymity on qualitative type statements made in the discussion. In some cases it is also possible to provide the privilege of allowing the respondents to choose when they wish to use pen-names and when they wish to use their real name. The more the individuals know one another and have a history as a "social" group, the more likely that good results will result from allowing participants to freely choose to use their real names, pen-names, or anonymity on qualitative type information.

There have been studies of computer based message systems which have attempted to conclude that the use of anonymity leads to "flaming" and antagonism (Kiesler, Siegel, and McGuire, 1984). Most of these observations have been based upon studying student groups who have no prior history or knowledge about one another. Flaming and disinhibition have not been problems among groups that already have a social history or social structure. When utilizing a computer based system with groups who are not familiar with one another, it may be important to provide a separate conference devoted to socializing among the group members. This would serve the same purpose as coffee breaks serve for co-located groups that work together. In the computer based communication environment, it has been observed that social-emotional exchanges are helpful in facilitating consensus development and eliminating potential misunderstanding (Hiltz, Johnson, and Turoff, 1986).

Anonymity for voting and estimates of subjective quantitative information is probably desirable to maintain in most circumstances. However, it is desirable that the coordinator for a Delphi exercise on the computer system be able to identify people with extreme votes or estimates. A Delphi coordinator should have no vested interest in the outcome and should be in a facilitation role. The facilitator may feel it is desirable to encourage individuals with extreme positions to explain them. Sometimes the observation that one is in a minority position can negatively affect participation unless there is such encouragement.

In some cases it may be desirable to allow voter identification. For example, in the final steps of a budget allocation task, it could be felt that everyone should assume final accountability for the recommended decision. Even in face-to-face committees, committee reports where no identified individuals assume responsibility have sometimes led to a lack of group commitment when it comes to implementing the results. Also, when no one is accountable, one can sometimes get more risky recommendations than would otherwise result. This decision must be based upon the nature of the application and the group. In any case, the identification of a member's voting position should only apply to the final evaluation phase of a group process.

3.4 MODERATION AND FACILITATION

In Computer Mediated Communication Systems, aside from message systems, if one wants to conduct group oriented communications, there is still a basic need for moderation and facilitation, just as in face-to-face meetings. However, the nature of leadership in the online environment is different from that in the face to face environment.

In the online environment it is much easier to separate the role of process facilitation from that of content leadership. It is also quite easy to develop a number of different leaders for different areas of a problem.

In the paper and pencil Delphi every contribution first goes to the coordinator of the exercise and then is integrated into a single summary provided to all of the participants. Clearly, in the computer based environment, this is not necessary. Whether or not given contributions need to be screened ahead of time is a function of the application and the nature of a particular contribution. Since the individual members can update themselves on what is new before making a contribution, the amount of duplication is minimized in a computer based Delphi.

For example, it may be desirable to hold certain types of contributions until the group is at a point in the deliberations where they are ready to deal with them. Also, information such as voting results should not be provided until a sufficient number of votes about an item have been accumulated. In situations dealing with very strong controversies, it may be necessary to screen and edit the wording of certain contributions to try to minimize emotional biases and tactics such as name calling and insulting remarks.

While a lot of material in an online Delphi can be delivered directly to the group, the specific decisions on this still need to be made by the person or team in control of the Delphi process. In Computer Mediated Communications, the activity level and actions of a conference moderator can be quite critical to the success of an asynchronous conference and specific guidelines for moderators can be found in the literature (Hiltz, 1984).

There have been many Delphis where the material is summarized based upon the breakdown of the respondents into various specialized expert subgroups or differing interests and perspectives. In the computer based environment it becomes possible to consider multiple group environments. This is where there are separate communication structures or separation of the respondents into sëparate Delphi groups. On top of this

structure would be a higher level one that synthesizes or filters out the reduced set of information necessary to pass between the groups. This does lead to the possibility of very large populations of respondents engaged in common task objectives. A practical example of this is multiple industrial standards groups which must be informed of what is arising from other groups that impacts on their considerations, but do not need to be involved in details of the subgroup deliberations in other areas.

There are many Delphi applications where respondents actually engage in taking on roles (e.g Stakeholder Analysis, Linstone, 1984) to deal with certain situations. This requires moderator supervision and direction. Associated with role playing is the employment of gaming situations where there may be groups in competition with one another and communication is regulated by the "game director" (Hsu, 1989). In the area of policy analysis it could be very productive to allow the subgroups that have agreement about a given resolution to have a private open conference where they can discuss the best possible responses to the material in the main Delphi as a private subgroup. It is also clear that subgroups could be formulated dynamically based upon the content of what is taking place.

Multiple group Delphis in a computer environment is a relatively new potential and there are no hard and fast rules for setting up communication structures in this area. As group oriented Computer Mediated Communication Systems become more widely used, there will be much opportunity to experiment with this relatively new opportunity for structuring communications at both the inter and intra group level.

3.5 STRUCTURE

The heart of a Delphi is the structure that relates all the contributions made by the individuals in the group and which produces a group view or perspective. In a computer based Delphi, the structure is one that reflects continuous operation and contributions. This is somewhat different than the paper and pencil mode where the structure must be divided into three or more discrete rounds. As an example, we will describe potential transformations of two simple structures that have often been utilized in paper Delphis, for use in a computerized environment.

3.5.1 The Policy Delphi

The first example is the Policy Delphi (Turoff, 1970). This is an interesting Delphi structure in that its objective is not to produce a consensus, but to expose the strongest pro and con arguments about differing resolutions of a policy issue. It is a form of policy analysis that provides a decision maker the strongest arguments on each side of the issue. Usually one utilizes as respondents individuals who have the strongest opposing views.

The structure of a Policy Delphi is very simple.

Policy Delphi Structure

TYPE OF ITEM	VOTING SCALES	RELATIONSHIPS
Resolution	Desirability Feasibility	Alternatives
Argument	Importance Validity	Pro or con to a given resolution Opposing to other arguments

In the above structure any respondent in the Delphi is free to add a possible resolution (solution) to the basic policy issue, or to make a pro or con argument about one or more of the listed possible resolutions. He or she can do this at any time. Also, the respondent can vote at any time on the two types of voting scales associated with either of the item types. Individuals may also choose to change their vote on a given item at any time. In this structure the two scales are needed to highlight situations where policy resolutions might be rated in such categories as desirable but infeasible, and arguments may be

rated as important but invalid (others might believe it). When making additions of a qualitative nature, participants must also indicate how that addition is related to the existing items.

The computer's role in the above process is to organize everything so that the individual can follow what is going on and obtain a group view:

Provide each member with new items that they have not yet seen.

Tally the votes and make the vote distribution viewable when sufficient votes are accumulated.

Organize a pro list and a con list of arguments about any resolution.

Allow the individual to view lists of arguments according to the results of the different voting scales (e.g. most valid to least valid arguments).

Allow the individual to compare opposing arguments.

Provide status information on how many respondents have dealt with a given resolution or list of arguments.

The role of the Delphi Coordinator or human facilitator is very minimum in such a well defined structure. The software powers or special privileges that such an individual needs are:

Being able to freeze a given list when it is felt there are sufficient entries to halt contributions, so as to focus energies on evaluation of the items entered to that point in time.

Being able to edit entries to eliminate or minimize duplications of resolutions or arguments.

Being able to call for final voting on a given item or set of items.

Being able to modify linkages between items when appropriate.

Reviewing data on participation so as to encourage participation via private messages.

It is possible to also develop rules to allow the computer to handle some of the above functions. But in terms of today's technology, these functions are still better handled by a human. A group using this structure for the first time should go through a training exercise. The Policy Delphi structure can be designed to be fairly easy to learn and utilize. The use of graphics to support visualization of the structure of the discussion can also be helpful.

The Policy Delphi structure was first implemented in paper and pencil in 1970 and was later implemented in two separate computer versions (Turoff, 1972; Conklin and

Begeman, 1987). It should be noted that the structure of relating items in a Policy Delphi may also be viewed as a representation of a specialized or tailored Hypertext system (Conklin, 1987; Nelson, 1965). Most Delphi designs, when translated to a computer environment, do depend upon semantic relationships among items being established and are utilized for browsing and presenting content oriented groupings of the material. A generalized approach to supporting Delphi relationships within a Hypertext environment may be found in the literature (Rao & Turoff, 1991; Turoff, Rao, and Hiltz, 1991).

Most Delphi structures can be considered to be types of items (i.e. nodes) which have various relationships (i.e. links) to one another. Therefore, it is possible to view a specific Delphi as a particular instance of a Hypertext system. Hypertext is the view of text fragments in a computer as the nodes within a graph or web of relationships making up a body of knowledge. Hypertext functionality is therefore useful for the support of automated Delphi processes.

3.5.2 The Trend Model

This Delphi involves first choosing a specific trend of concern to the group. For example, this might be deaths from AIDS or the amount of life extension expected from a particular treatment. One might include in a single study a set of related trend variables. For the purpose of this explanation we will focus on one trend.

The individual respondents are asked first to make a projection of where they think the time curve will go in the next five years. Then they are asked to list the assumptions they are making and any uncertainties they have. Assumptions are things they think will occur over that time frame and which impact on determining this trend. Uncertainties are things they do not think will occur, but if they did, they would cause changes in estimates of where the trend will go.

Since some peoples' uncertainties are other's assumptions, these are compiled into a list of "possible" assumptions and every individual is asked to vote on each possible assumption according to validity. To accomplish this validity estimation the group may be provided with an anchored interval scale which varies, for example, from "definitely true" to "definitely false," with a mid-point of "maybe." The resulting list of assumptions is automatically reordered by the group validity judgement. The ones the group agrees on as valid or invalid are set aside, and the subsequent discussion focuses

on the assumptions that have an average vote of "maybe". The analysis of the voting has to point out which "maybe" votes result from true uncertainty on the part of the respondents, and which result from wide differences in beliefs between subgroups of respondents.

Clearly in the computer environment, this process of listing, voting, and discussing the assumptions can take place on a continuous basis. The voting serves to quickly eliminate from the discussion those items on which the group agrees. The remaining uncertain items usually are divided into two types: 1) those which can be influenced (e.g. improvements in knowledge about the proper use of condoms), 2) those that cannot be influenced (e.g. hospital facilities in the short term).

In the final stage, after the list has been completed and evaluated, the participants are asked to re-estimate their earlier trend estimate. One could observe that a statistical regression analysis might have produced a similar trend curve. However, the application of such a mathematical technique will not produce the qualitative model that represents the collective judgement of all the experts involved. It is that model which is important to understanding the projection and what actions can be taken to influence changes in the trend or in understanding the variation in the projection of the trend.

There is practically no planning task where the above trend analysis structure is not applicable. In the medical field, for example, considering examining trend curves for the occurrences of certain medical problems and the impact of various treatments is rather broadly applicable. This particular structure has been utilized in a significant number of corporate planning exercises. With graphic capabilities on workstations, it would be quite easy to implement in a computerized version. A similar structure may be applied to qualitative trends made up of a time series of related discrete events. An example would be AIDS cases triggering specific legal rulings and particular ethical dilemmas.

The above two examples were chosen because they are fairly simple and straightforward. However, there are literally tens of different Delphi structures that have been demonstrated in the paper and pencil environment (Linstone and Turoff, 1975) and are quite transferable to the computer based environment. Many of these require the ability to utilize graphics to view the complexity of relationships among concepts. Others require extended facilities to utilize generalized Hypertext structures. However, one of the most significant potentials for the automation of Delphi is the incorporation

of real time analysis aids for the interpretation and presentation of the subjective information produced in a Delphi exercise. This will be treated in a following section. It should also be clear from the above examples that there are certain fundamental tools that apply across a wide range of Delphi structures. The ability of a group to contribute to building a specific list, to be able to apply specific voting capabilities, and to be able to sort the list by voting results, represents a set of general tool capabilities. This is the approach we have taken in the development of the EIES 2 system (Turoff, 1991) at NJIT to support a wide variety of applications such as Group Decision Support, Delphi Design, Project Management, and Education (Hiltz, 1986, 1990).

EIES 2 is a general purpose Computer Mediated Communication System that provides many features whereby an individual moderating a specific conference can tailor the group process. The moderator of a given conference can create, at any point in the discussion, an "activity" that may be attached to a comment. These activities accomplish different specialized functions such as list collection and voting. However, the interface to all these activities is the same in the sense that the same basic generic commands apply to any activity. For example, one may "Do" the activity to make changes to it or "View" the results of the activity, regardless of what type of activity it is. The conference moderator has the authority to introduce these activities whenever he or she feels they fit within the current discussion. Also, the moderator may choose to allow or not allow the facilities of anonymity for a given activity or conference.

EIES 2 also provides a general notifications capability that can be tailored to notify the participants in a group process whenever any action occurs of which they need to be made aware. For example, a notification may let the members of a Delphi know when the votes on a specific item are sufficient to allow viewing of the resulting distribution.

EIES 2 is constructed so that any programs or analysis routines developed in any language within the context of the UNIX operating system or a TCP/IP network can be integrated or made available through the EIES 2 interface. The major facility to allow a Computer Mediated Communication System to enhance Delphi processes is to provide alternative structures in the form of a collection of group support tools. The system must also include the privileges for a facilitator or group leader to decide on the dynamic incorporation of these tools in the group process.

3.6 ANALYSIS

A principal contribution to the improvement of the quality of the results in a paper and pencil Delphi study is the analysis that the design and coordination team can perform on the results of each round. This analysis has a number of specific objectives:

Improve the understanding of the participants through analysis of subjective judgements to produce a clear presentation of the range of views and considerations.

Detect hidden disagreements and judgmental biases that should be exposed for further clarification.

Detect missing information or cases of ambiguity in interpretation by different participants.

Allow the examination of very complex situations that can only be summarized by analysis procedures.

Detect patterns of information and of sub-group positions.

Detect critical items that need to be focused upon.

To accomplish the above, there are a host of analysis approaches that come from many different fields. Many of these are amenable to implementation as real time computer based support to a continuous Delphi process conducted via a Computer Mediated Conferencing System. We will briefly address here some of the most significant types of these methods for supporting Delphi applications.

3.6.1 Scaling Methods

Scaling is the science of determining measuring instruments for human judgement. Clearly, one needs to make use of appropriate scaling methods to aid in improving the accuracy of subjective estimation and voting procedures. While most of these methods were originally developed to measure human judgement, they are easily adaptable, in many cases, to providing feedback to a Delphi group on the consequences of the judgements being made by the individuals.

For example, in many cases the appropriate judgement we wish to solicit from an individual is a ranking (i.e. ordinal scale measurement) of individual items. It is comparatively more accurate to ask individuals to rank order items, such as objectives or goals, than to ask for interval or ratio measures. A person can estimate that a particular goal is more important than another one; however, how much more important it is much more difficult to estimate consistently among a group of individuals. However, a scaling method such as Thurstone's Law of Comparative Judgement

(Torgenson, 1958) can transform individual ranking judgements and produce analytically a group result which is an interval scale rather than a rank ordered scale. Providing the group the results in terms of this interval scale allows the individuals to detect in a much more reliable manner the extent to which certain objectives are clearly distinct from other objectives, and which are considered in closer proximity. Merely providing an averaging of the ranking scale does not contribute this added insight to the group as a whole. Furthermore, standard averaging approaches can lead to inconsistencies in group judgements (i.e. Arrow's Paradox). This can occur when there are disagreements underlying the averaging and when there is a lack of appropriate "anchoring" of the scales.

Standard correlation analysis approaches can be utilized to determine if there are subgroups or patterns of agreement and disagreement that exist across different issues or judgements made in the Delphi exercise. Do the people who feel a certain way about an issue feel the same way about another issue? This type of analysis should, in most cases, be provided first to the facilitator, and that person should decide which relationships need to be passed back to the group. In many Delphis, there are identified sub-groups. A Delphi might comprise people from different disciplines. Do the administrators, researchers, lawyers, insurers, and practitioners have differences in viewpoint that are based upon the perspective they take on a new medical treatment? The utility of these insights needs to be evaluated by the facilitator in the context of the application. With groups that work together over a long term, it might be desirable to provide such an analysis in terms of direct feedback without facilitator intervention.

Scaling methods span a wide range of techniques, from fairly simple and straightforward to fairly sophisticated. An example of a sophisticated approach is Multi-Dimensional Scaling (Carroll and Wish, 1975). MDS allows subjective estimates of similarity between any two objects to be translated into a relative position in a Euclidean space. It provides, in essence, N dimensional interval scaling of similarity estimates. The number of meaningful dimensions found suggests the number of independent dimensional factors underlying the way both the individuals and the group are viewing the similarity among objects. By looking at the alternative two dimensional projections, it is possible to arrive at an understanding of what the dimensional factors are.

The process by which one would use MDS in a Delphi would be to ask for the similarities and provide back the graphical layouts of the alternative dimensions. The respondents would then be asked to try to determine what these dimensions mean or represent. The result is a very powerful technique for potentially exposing the hidden factors a group is using to make judgements about similarities. The question of similarity is one that can be applied to a very wide range of object types, e.g. goals, products, countries, relationships, jobs, criteria, etc. MDS may also be viewed as a form of Cluster Analysis, and many methods in Cluster Analysis (Anderberg, 1973) can also be usefully applied to analyzing the subjective comparison judgements made by Delphi respondents.

When a group is using voting and estimation structures over a long period so that they make judgements about a growing number of similar situations, it is possible to consider the introduction of "scoring" methods (Dalkey, 1977), into the Delphi process. Given later feedback upon the accuracy of estimates or the quality or success of a given judgement, it is possible to consider feedback to estimators on their degrees of "accuracy" or possible biases due to factors such as conservatism. At the point where there are individuals utilizing Delphi techniques on a continuous basis, it will be possible to conduct the sorts of investigations needed to develop this particular area as a decision aid.

Designing a Delphi, whether via paper and pencil or on the computer, does include the process of designing a survey. As such, all the guidelines on good survey design and all the analysis methods that have been developed for analyzing survey data are potentially applicable to a Delphi. There is, however, a fundamental difference in objectives, which determines how one employs a given method, and whether it is applicable in a given situation.

Most scaling methods were evolved to aid in making an assessment of a human judgement with the premise that one is measuring a stable and constant quantity. One's intelligence or personality would not be affected or changed as a part of the measurement process. The goal is to discover biases and inconsistencies and to produce more accurate measurements. In the Delphi process, however, we are interested in informing the respondents about what they are really saying, and how it compares to the group as a whole. We are also interested in promoting changes in viewpoints and the other items we measure, if it will promote reaching a superior group view of the

situation. We are also interested in detecting and exposing hidden factors or relationships of which the group may not be completely aware. With this in mind, one has to take special care that the use of these analysis methods does not convey a false impression of finalization in a group view.
Related to scaling is the area of Social Choice Theory, which provides alternative methods for the summarization of voting processes (Hogarth, 1977). The use of multiple methods of viewing the summarization of a given voting process can be useful in preventing a group from placing an over emphasis on a single voting result.
Probably the most important single consideration in the past that has prevented the incorporation of many of the approaches discussed here is the difficulty of educating the respondent in the interpretation of the method when the respondent is involved in only one short term Delphi process. With the potential that Computer Mediated Communications offers for long term continuous use by groups, it is now possible to consider incremental training for individuals to gain an understanding of the more sophisticated methods.
With the appropriate use of scaling methods it becomes possible to establish that individuals will mean the same thing when they use terms like: desirable, very desirable, likely, unlikely, agree, strongly agree, etc. It becomes possible to determine which alternatives are truly similar and which are distinctly different. Scaling methods, in essence, serve the objective of eliminating ambiguity in the judgmental and estimation process of a group.

3.6.2 Structural Modeling

The term Structural Modeling (Lendaris, 1980; Geoffrion, 1987) has come to represent a host of specific methods that have the objective of allowing an individual to express a large set of independent relationships and judgements which the given method utilizes to produce a "whole" model of the "system" being described. In computer terms, these are methods that allow a user to build a model of a situation without having to program or go through the use of experts in modeling and simulation. These methods vary from ones that provide a simple static relationship model (e.g. Interpretive Structural Modeling, Warfield, 1974), to more dynamic probabilistic and time varying models

NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ARCHIEVE PROGRAM

COM 400 PROJECT

ŞEFKİ KOLOZALİ (20010333)

NEBİLE KIRMIZILAR (20000325)

SUBMİTTED TO: ÜMİT SOYER

NICOSIA 2005

ACKNOWLEDGEMENT

We would like to thank to our supervisors Umit Soyer , without whom this project would have not been possible. Whose words of encouragement kept us doing our project. His invaluable knowledge for the project has made our taking keen interest in our project.

Specially we would like to thank Hasan Yucel who helped us so much in doing our project. We would also like to thank Caner Cakir for his helped, he was always near us and helped us. Also thank for all our friend to be belive us and give as moral: Yazen Al Shawneh , Sule Akkurt ,Mustafa Sevki Hizal ,Duygu Ozsen and Ahmet Unler. Being with them make our time full of fun. They encouraged us allot in completing our project. We want to thank them as they contributed their time and provided very helpful suggestions.

Finally, we want to thank our family, especially our parents. Without their endless support and love for us, we would never achieve our current position.

ABSTRACT

The archive system is the principles of organization and description. It use for to find our documents more easy than looking step by step all the documents. In our project we did an archive program to fine documents and images more easy without looking anywhere just write name of what you are looking and search so it will be more easy to find what we are looking. With out move your place you can find what ever you are looking. But all of this making with controlling. But we have to know how we will fine. Research is work that tries to discover facts about something. if there is resemblance between two things there similar to each other. if you research something ,you try to discover facts about it.

TABLE OF CONTENTS

ACKNOWLEDGEMENT i

ABSTRACT ii

TABLE OF CONTENTS iii

INTRODUCTION viii

CHAPTER 1 1

THE ARCHITECTURE OF DATABASE APPLICATIONS 1

1.1 ARCHITECTURE OF DATABASE 1

1.1.1 RELATIONAL DATABASE CONCEPTS 1

1.1.1.1 DATABASE TABLES 1

1.1.1.1.1 CHARACTERISTICS OF RELATIONAL TABLES 2

1.1.1.1.2 WHAT DO YOU KEEP IN EACH TABLE? 3

1.1.1.2 RELATIONS BETWEEN TABLES 3

1.1.1.2.1 TABLES HAVE A PRIMARY KEY 4

1.1.1.2.1.1 EXAMPLES 4

1.1.1.2.2 TABLES ARE RELATED BY FOREIGN KEYS 4

1.1.1.2.2.1 EXAMPLE 5

1.1.1.3 OTHER DATABASE OBJECTS 5

1.1.2 THE PIECES OF A DATABASE SYSTEM 7

1.1.3 HOW THE PIECES FIT TOGETHER 9

1.1.3.1 PERSONAL APPLICATIONS AND EMBEDDED DATABASES 9

1.1.3.2 CLIENT/SERVER APPL. AND MULTI-USER DATABA 10

1.1.4 MULTI-TIER COMPUTING ARCHITECTURE 11

1.1.5 USING MULTIPLE DATABASES 12

1.1.5.1 RUNNING DATABASES ON A SINGLE DATABASE SERVER ... 12

1.1.5.2 ACCESSING DATA IN OTHER DATABASES 13

1.2 ABOUT DBASE AND PARADOX 13

1.2.1 DBASE IV TABLE SPECIFICATION 13

1.2.2 DBASE V TABLE SPECIFICATIONS 14

1.2.3 DBASE FIELD TYPES 14

1.2.4 PARADOX STANDARD TABLE SPECIFICATIONS 15

1.2.5 PARADOX 5 TABLE SPECIFICATIONS 16

1.2.6 PARADOX 7 AND ABOVE TABLE SPECIFICATIONS 17

1.2.7 PARADOX FIELD TYPES 17

1.2.8 ABOUT OUR DATABASE **19**

CHAPTER 2 22

OPERATION SYSTEM 22

2.1 OPERATING SYSTEM 22

2.2 OPERATING SYSTEM TYPES 22

2.2.1 MULTI-USER 23

2.2.2 MULTIPROCESSING 23

2.2.3 MULTITASKING 23

2.2.4 MULTITHREADING 23

2.3 INTRODUCTION TO UNIX: 24

2.3.1 OBJECTIVES 24

2.3.2 WHAT IS AN OPERATING SYSTEM? 24

2.3.3 A BRIEF HISTORY OF UNIX 26

2.3.4 ARCHITECTURE OF THE LINUX OPERATING SYSTEM 29

2.4 AN INTRODUCTION TO LINUX 31

2.4.1 WHAT IS LINUX? 31

2.4.2 WHY USE LINUX? 32

2.4.2.1 CONFIGURABILITY 32

2.4.2.2 CONVENIENCE 33

2.4.2.3 STABILITY 33

2.4.2.4 COMMUNITY 33

2.4.2.5 FREEDOM 34

2.5 THE DIFFERENCES BETWEEN LINUX AND OTHER UNIX 34

CHAPTER 3 .... ***35***

DELPHI.. *35*

3.1.INTRODUCTION.. *35*

3.2. ASYNCHRONOUS INTERACTION *37*

3.3 ANONYMITY *40*

3.4 MODERATION AND FACILITATION *43*

3.5 STRUCTURE *44*

3.5.1 The Policy Delphi *45*

3.5.2 The Trend Model *47*

3.6 ANALYSIS.. *50*

3.6.1 Scaling Methods *50*

3.6.2 Structural Modeling.. *53*

3.7 DELPHI, EXPERT SYS., GDSS, AND COLLABORATIVE SYSTEMS..... 55

3.7.1 The Deductive Level of Disagreements *57*

3.7.2 The Inductive Level of Disagreements *58*

3.7.3 Goal and Value Disagreements *59*

3.8 CONCLUSION *60*

CHAPTER 4 .... **61**

ARCHIVE SYSTEM 61

4.1 Basic concepts and principles of archives and records management 62

4.1.1 The Inter-relatedness of records 62

4.1.2 The central importance of context.. 62

4.1.3 The function of records as evidence 63

4.1.3.1 Recordness requires that records are: 63

4.1.3.1.1 Complete.. 64

4.1.3.1.2 'Fixed' 64

4.1.3.1.3 Organic 64

4.1.3.1.4 Contextual 64

4.1.3.1.5 Authoritative/'Official' 65

4.1.3.1.6 Unique 65

4.1.3.1.6.1. The principle of provenance 65

4.1.3.1.6.2. The principle of original order 65

4.1.3.1.6.3. The chain of responsible custody 67

4.1.3.1.6.3.1 What goes on in the archives? 67

4.1.3.1.6.3.2 CADSS 67

4.1.3.1.6.3.3 CONTROL 68

4.1.3.1.6.3.4 ACCESSIBILITY 69

4.1.3.1.7 DISPOSAL 71

4.1.3.1.8 STORAGE 72

4.1.3.1.9 Check Answer 73

4.1.3.1.10 SUSTAIN 75

4.1.3.1.11 Recordkeeping regimes 77

4.1.3.1.12 Where we work: The recordkeeping context 77

4.1.3.1.12.1 Public sector placements 77

4.1.3.1.12.2 Private sector placements 78

4.1.3.1.13 Type and functional emphasis 78

4.1.3.1.14 Accountable recordkeeping regimes at macro level. 79

4.1.3.1.15 Recordkeeping rewards 82

4.1.3.1.16 Recordkeeping and related professions 82

4.1.3.1.16.1 What's the difference? 82

4.1.3.1.17 Recordkeeping and specialist recordkeepers 83

CHAPTER 5 84

ARCHIVE PROGRAM 84

CONCLUSION 157

REFERNCES..157

INTRODUCTION

Archive system is main base registration function of office management for extension. For correct registration of incoming and outgoing files or documents you need always true programming system. Our program is using computer for registration sending and receiving documents from selected office. Documents are enter to the Archive system with their attachments. Any one can easily reach all of enclosed file or documents.

Our first chapter is all about explaining Architecture of database systems. About relational database's concepts, the pieces of database systems, how the pieces fit together, multi-tier computing architecture and multiple databases, dbase and paradox.

Our second chapter is all about giving details of operation systems. Which are Unix, Linux and the differential of Unix and Linux.

Our third chapter is all about Delphi. Introduction of Delphi, asynchronous interaction ,anonymity and so on.

Our fourth chapter we explained about the archive systems. Basic concepts and principles of archives and how its work.

Our last fifth chapter is about the our program archive program how its wo.how we can use it and so on.

CHAPTER 1

The Architecture of Database Applications

1.1 Architecture of database

1. Relational database concepts
2. The pieces of a database system
3. How the pieces fit together
4. Multi-tier computing architecture
5. Using multiple databases
6. About dbase

1.1.1 Relational database concepts

A relational database-management system (RDBMS) is a system for storing and retrieving data, in which the data is organized into interrelated tables.

SQL Anywhere Studio provides two relational database systems. Adaptive Server Anywhere is the primary, full featured RDBMS, with a multitude of uses, from a network database server hosting many clients to a compact embedded database. UltraLite is a small-footprint relational database. The UltraLite deployment technology allows you to use Adaptive Server Anywhere features on even the smallest of devices.

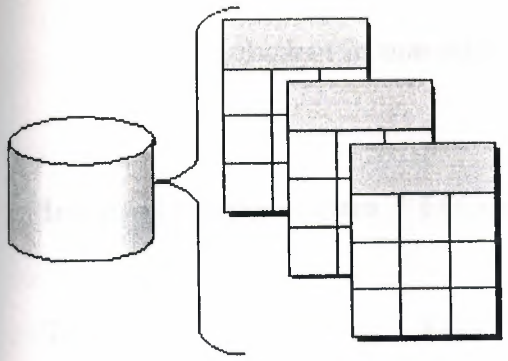
1.1.1.1 Database tables

1.1.1.2 Relations between tables

1.1.1.3 Other database objects

1.1.1.1 Database tables

In a relational database, all data is held in tables, which are made up of rows and columns.



Each table has one or more columns, and each column is assigned a specific data type, such as an integer, a sequence of characters (for text), or a date. Each row in the table has a single value for each column.

For example, a table containing employee information may look as follows:

emp_ID	emp_lname	emp_fname	emp_phone
10057	Huong	Zhang	1096
10693	Donaldson	Anne	7821

1.1.1.1.1 Characteristics of relational tables

The tables of a relational database have some important characteristics:

- There is no significance to the order of the columns or rows.
- Each row contains one and only one value for each column, or contains NULL, which indicates that there is no value for that column.
- All values for a given column have the same data type.

  The following table lists some of the formal and informal relational database terms describing tables and their contents, together with their

equivalent in non-relational databases. This manual uses the informal terms.

Informal relational term	Formal relational term	Non-relational term
Table	Relation	File
Column	Attribute	Field
Row	Tuple	Record

1.1.1.1.2 What do you keep in each table?

Each table in the database should hold information about a specific thing, such as employees, products, or customers.

By designing a database this way, you can set up a structure that eliminates redundancy and the possible inconsistencies caused by redundancy. For example, both the sales and accounts payable departments might enter and look up information about customers. In a relational database, the information about customers is stored only once, in a table that both departments can access.

1.1.1.2 Relations between tables

You use primary keys and foreign keys to describe relationships between the information in different tables. Primary keys identify each row in a table uniquely, and foreign keys define the relationships between rows in different tables.

Primary keys and foreign keys let you use relational databases to hold information in an efficient manner, without redundancy.

1.1.1.2.1 Tables have a primary key

1.1.1.2.2 Tables are related by foreign keys

1.1.1.2.1 Tables have a primary key

Each table in a relational database should have a primary key. The primary key is a column, or set of columns, that uniquely identifies each row. No two rows in a table may have the same primary key value

.

1.1.1.2.1.1 Examples

In the sample database, the employee table stores information about employees. It has a primary key column named emp_id, which holds a unique ID number assigned to each employee. A single column holding an ID number is a common way to assign primary keys, and has advantages over names and other identifiers that may not always be unique.

A more complex primary key can be seen in the sales_order_items table of the sample database. The table holds information about individual items on orders from the company, and has the following columns:

- id An order number, identifying the order the item is part of.
- line_id A line number, identifying each item on any order.
- prod_id A product ID, identifying the product being ordered.
- quantity A quantity, displaying how many items were ordered.
- ship_date A ship date, displaying when the order was shipped.

A particular sales order item is identified by the order it is part of, and by a line number on the order. These two numbers are stored in the id and line_id columns. Items may share a single id value (corresponding to an order for more than one item) or they may share a line_id number (all first items on different orders have a line_id of 1). No two items share both values, and so the primary key is made up of these two columns.

1.1.1.2.2 Tables are related by foreign keys

The information in one table is related to that in other tables by foreign keys.

1.1.1.2.2.1 Example

The sample database has one table holding employee information and one table holding department information. The department table has the following columns:

- dept_id An ID number for the department. This is the primary key for the table.
- dept_name The name of the department.
- dept_head_id The employee ID for the department manager.

To find the name of a particular employee's department, there is no need to put the name of the employee's department into the employee table. Instead, the employee table contains a column holding a number that matches one of the dept_id values in the department column.

The dept_id column in the employee table is called a foreign key to the department table. A foreign key references a particular row in the table containing the corresponding primary key.

In this example, the employee table (which contains the foreign key in the relationship) is called the foreign table or referencing table. The department table (which contains the referenced primary key) is called the primary table or the referenced table.

1.1.1.3 Other database objects

There is more to a relational database than simply a set of related tables. You will also find the following objects in a relational database:

- Indexes Indexes allow quick lookup of information. Conceptually, an index in a database is like an index in a book. In a book, the index relates each indexed term to the page or pages on which that word appears. In a database, the index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored.

Indexes are an important design element for high performance. You must usually create indexes explicitly, but indexes for primary and foreign keys and for unique columns are created automatically. Once created, the use of indexes is transparent to the user.

- Views Views are computed tables, or virtual tables. They look like tables to client applications, but they do not hold data. Instead, whenever they are accessed, the information in them is computed from the underlying tables.

  The tables that actually hold the information are sometimes called base tables to distinguish them from views. A view is defined with a SQL query on base tables or other views.

- Stored procedures and triggers These are routines held in the database itself that act on the information in the database.

  You can create and name your own stored procedures to execute specific database queries and to perform other database tasks. Stored procedures can take parameters. For example, you might create a stored procedure that returns the names of all customers who have spent more than the amount that you specify as a parameter in the call to the procedure.

  A trigger is a special stored procedure that automatically fires whenever a user updates, deletes, or inserts data, depending on how you define the trigger. You associate a trigger with a table or columns within a table. Triggers are useful for automatically maintaining business rules in a database.

- Users and groups Each user of a database has a user ID and password. You can set permissions for each user so that confidential information is kept private and users are prevented from making unauthorized changes. Users can be assigned to groups in order to make the administration of permissions easier.

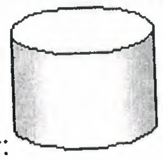
- Java objects You can install Java classes into the database. Java classes provide a powerful way of building logic into your

database, and a special class of user-defined data types for holding information.

All of these items together make up a relational database-management system (RDBMS); a system for storing and retrieving data, in which the data is organized into interrelated tables.

1.1.2 The pieces of a database system

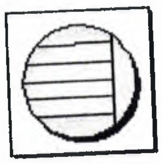
Relational database management systems contain the following pieces:

A database Data is stored in a database. In diagrams in the documentation, a database is indicated by a cylinder: 

An Adaptive Server Anywhere database is a file, usually with an extension of .db. Adaptive Server Anywhere includes a sample database for you to work with: this is the file asademo.db in your Adaptive Server Anywhere installation directory.

A database server The database server manages the database. No other application addresses the database file directly; they all communicate with the database server.

In the documentation, a database server is indicated as follows:



Adaptive Server Anywhere provides two versions of its database server: the personal database server and the network database server. In addition to the features of the personal server, the network database server also supports client/server communications across a network, while the personal database server can accept connections only from applications running on the same machine. The request-processing engine is identical in both servers.

A programming interface Applications communicate with the database server using a programming interface. You can use ODBC, JDBC, OLE DB, Sybase Open Client, or Embedded SQL.

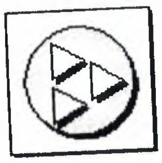
Many application development tools provide their own programming environment that hides the details of the underlying interface. For example, if you develop an application using Sybase PowerBuilder, you never have to make an ODBC function call. Nevertheless, behind the scenes each of these tools is using one of the programming interfaces.

The programming interface provides a library of function calls for communicating with the database. For ODBC and JDBC, the library is commonly called a **driver**. The library is typically provided as a shared library on UNIX operating systems or a dynamic link library (DLL) on PC operating systems. The JDBC interface uses the Sybase jConnect driver, which is a zip file of compiled Java classes.

A client application Client applications use one of the programming interfaces to communicate with the database server.

If you develop an application using a rapid application development (RAD) tool such as Sybase PowerBuilder, you may find that the tool provides its own methods for communicating with database servers, and hides the details of the language interface. Nevertheless, all applications use one of the supported interfaces.

In diagrams in the documentation, a client application is indicated by the following icon:



UltraLite database servers are custom-generated for each UltraLite application, and are part of the application itself. An UltraLite application together with its database server is indicated as follows:

1.1.3 How the pieces fit together

Database applications can connect to a database server located on the same machine as the application itself, or in the case of the network database server, on a different machine. In addition, with Adaptive Server Anywhere you can build distributed databases, with physically distinct databases on different machines sharing data.

1.1.3.1 Personal applications and embedded databases
1.1.3.2 Client/server applications and multi-user databases

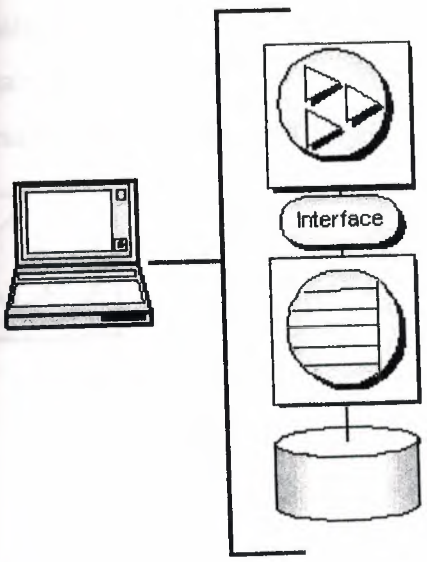
1.1.3.1 Personal applications and embedded databases

You can use Adaptive Server Anywhere to build a complete application and database on a single computer. In the simplest arrangement, this is a standalone application or personal application: it is self-contained with no connection to other databases. In this case, the database server and the database can be started by the client application, and it is common to refer to the database as an embedded database. As far as the end user is concerned, the database is a part of the application. When a database server is used as an embedded database, it is sometimes called a database engine.

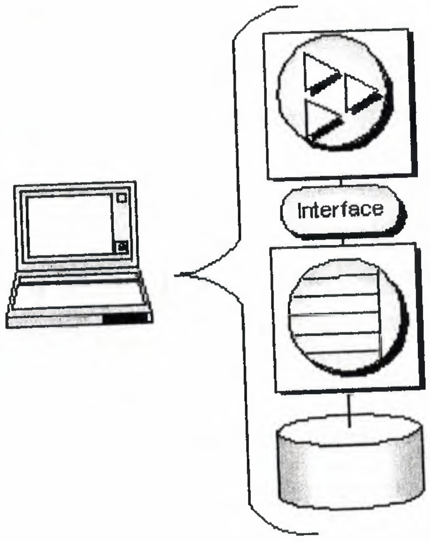
Many relational database management systems require experienced staff for administration. A characteristic of embedded databases is the ability to run entirely without administration.

The Adaptive Server Anywhere personal database server is generally used for standalone applications. A client application connects through a programming interface to a database server running on the same machine:

Standalone applications have the following architecture, with a client application connecting through a programming interface to a database server running on the same machine:



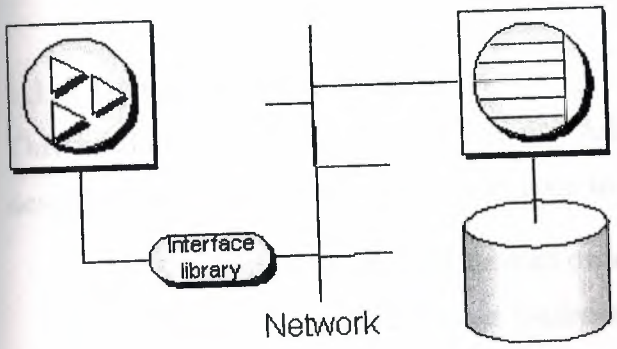
The Adaptive Server Anywhere personal database server is generally used for standalone applications, although you can also use applications on the same machine as the network server.



1.1.3.2 Client/server applications and multi-user databases

You can use Adaptive Server Anywhere to build an installation with many applications running on different machines, connected over a network to a single

database server running on a separate machine. This is a client/server or multi-user database environment, and has the following architecture. The interface library is located on each client machine.



In this case, the database server is the Adaptive Server Anywhere network database server, which supports network communications.

For a client application to work in a client/server environment, you need only identify the server to which it should connect.

For a client application to work in a client/server environment, you need only identify the server to which it should connect.

1.1.4 Multi-tier computing architecture

In multi-tier computing, application logic is held in an application server, such as Sybase EAServer, which sits between the database server and the client applications. In many situations, a single application server may access multiple databases in addition to non-relational data stores. In the Internet case, client applications are browser-based, and the application server is generally a Web server extension.

Sybase EAServer stores application logic in the form of components, and makes these components available to client applications. The components may be PowerBuilder components, Java beans, or COM components.

Application servers can also provide transaction logic to their client applications—guaranteeing that sets of operations are executed atomically across multiple databases. Adaptive Server Anywhere is well-suited to multi-tier computing, and can participate in distributed transactions coordinated by Microsoft Distributed Transaction Coordinator.

Both Sybase Enterprise Application Server and Microsoft Transaction Server use DTC to provide transaction services to their client applications.

1.1.5 Using multiple databases

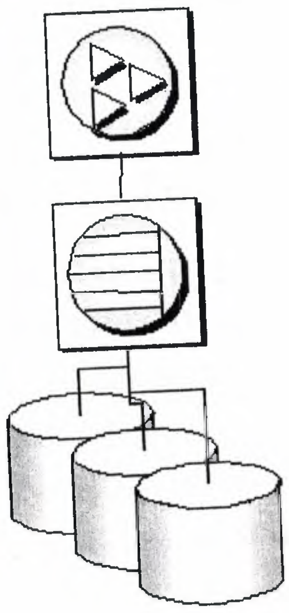
This section describes extensions to the Adaptive Server Anywhere architecture described above for the case where you want to use more than one database.

1.1.5.1 Running multiple databases on a single database server

1.1.5.2 Accessing data in other databases

1.1.5.1 Running multiple databases on a single database server

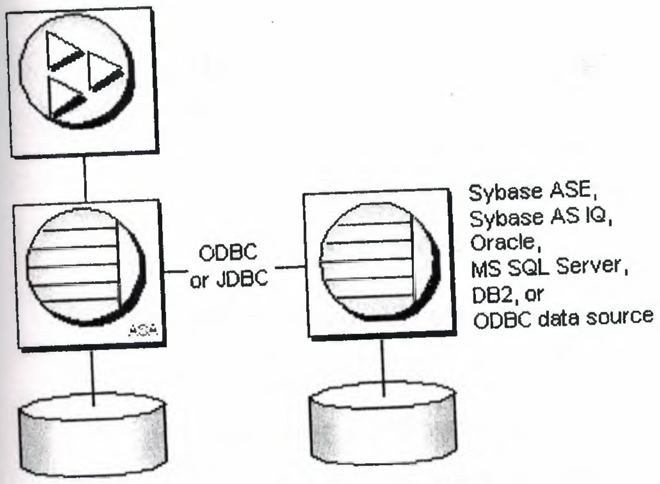
Both the Adaptive Server Anywhere personal database server and the network database server can manage several databases simultaneously. Each connection from an application must be to a single database, but an application can use separate connections to different databases, or a set of applications can work on different databases, all through the same database server.



Databases can be started when the database server is started, or by connecting to a database using the DatabaseFile connection parameter.

1.1.5.2 Accessing data in other databases

You can access databases on multiple database servers, or even on the same server, using the Adaptive Server Anywhere Remote Data Access features. The application is still connected to a single database as in the architecture diagrams above, but by defining remote servers, you can use proxy tables that exist on the remote database as if they were in the database to which you are connected.



1.2 About Dbase And Paradox

1.2.1 dBASE IV Table Specification

The dBASE IV table format was introduced in dBASE IV for DOS. Following are the specifications for dBASE IV tables.

2GB file size.

Two billion records per file.

A maximum of 255 fields per record.

Maintained indexes can have up to 47 indexes per file. Each index can be created using field expressions of virtually any combination, including conditional expressions of up to 255 characters per expression that result in an index of up to 100 bytes.

Unlimited nonmaintained indexes can be stored on disk. You can use up to 47 of them simultaneously.

1.2.2 dBase V Table Specifications

The dBASE V table format was introduced in dBASE V for Windows. Following are the specifications for dBASEA V tables.

- Up to one billion records per file.
- A maximum of 1,024 fields per record.
- Up to 32,767 bytes per record.
- Unlimited nonmaintained indexes can be stored on disk. You can use up to 47 of them simultaneously.
- Up to 10 master index files open per database. Each master index can have up to 47 indexes.
- Maintained indexes can have up to 47 indexes per file. Each index can be created using field expressions of virtually any combination, including conditional expressions of up to 255 characters per expression that result in an index of up to 100 bytes.

1.2.3 dBASE Field Types

Character (C)

dBASE III+, IV, and V field type that can contain up to 254 characters (including blank spaces). This field is similar to the Paradox Alpha field type.

Date (D)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V. dBASE tables can store dates from January 1, 100, to December 31, 9999. Paradox 5 tables can store from 12/31/9999 B.C. to 12/31/9999 A.D.

Float (F)

dBASE IV, and V floating-point numeric field type provides up to 20 significant digits.

Logical (L)

Paradox 5 and 7 and dBASE III+, IV, and V field type can store values representing True or False (yes or no). By default, valid entries include T and F (case is not important).

Memo (M)

Paradox 4, 5, and 7 as well as dBASE III+, IV, and V field. A Paradox field type is an Alpha variable-length field up to 256MB per field. dBASE Memo fields can contain binary as well as memo data.

OLE (O)

Paradox 1, 5, and 7 as well as dBASE V field type that can store OLE data.

Number (N)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V field type can store up to 15 significant digits -10307 to + 10308 with up to 15 significant digits.

dBASE number fields contain numeric data in a Binary Coded Decimal (BCD) format. Use number fields when you need to perform precise calculations on the field data. Calculations on number fields are performed more slowly but with greater precision than are calculations on float number fields. The size of a dBASE number field can be from 1 to 20. Remember, however, that BCD is in Paradox 5 and 7 only for compatibility and is mapped directly to the Number field type.

Short (S)

Paradox 3.5, 4, 5, and 7 field type that can contain integers from -- 32,767 through 32,767 (no decimal).

1.2.4 Paradox Standard Table Specifications

Also known as Paradox 4 table structure.

The Paradox standard table format was introduced in Paradox for DOS version 4. Other products that use the standard format include Paradox for DOS version 4.5, ObjectVision 2.1, and Paradox for Windows versions 1.0 and 4.5.

Earlier versions of the Paradox table type are referred to as the Compatible table type. In the BDE Configuration Utility, the level option for the Paradox driver dictates what default table type is created by Paradox for Windows. Use 3 for Compatible tables, 4 for Standard tables (the default). Following are the specifications for standard Paradox tables.

256MB file size limit if the table is in Paradox format and using a 4K block size.

Up to 255 fields per record.

Up to 64 validity checks per table.

A primary index can have up to 16 fields.

Tables can have up to 127 secondary indexes.

Up to two billion records per file. Because of the 256MB file size limit and other factors such as block size, however, the limit is much smaller. Tables of 190,000 records are easily achievable (and you can have more if you don't use up the 1,350-bytes-per-record limit for a keyed table). Tables with close to a million records are common.

Block size can be 1024, 2048, 3072, or 4096. Paradox stores data in fixed records. Even if part or all of the record is empty, the space is claimed. Knowing the interworkings can save you disk space. Paradox stores records in fixed blocks of 1024, 2048, 3072, 4096 in size.

After a block size is set for a table, that size is fixed, and all blocks in the table will be of that size. To conserve disk space, you want to try to get your record size as close to a multiple of block size as possible (minus 6 bytes, which are used by Paradox to manage the table).

Record size. 1,350 for keyed tables and 4,000 for unkeyed tables. When figuring out the size (the number of bytes or characters) of a table, remember that Alpha fields take up their size (for example, an A10 = 10 bytes), numeric field types take up 8 bytes, short number field types take up 2 bytes, money takes up 8, and dates take up 4 bytes.

Memos, BLOBs, and so on take 10 bytes plus however much of the memo is stored in the .DB. For example, M15 takes 25 bytes.

1.2.5 Paradox 5 Table Specifications

The Paradox 5 table format was introduced in Paradox for Windows version 5. Following are the specifications for Paradox 5 tables.

Up to two billion records per file.

File size is limited to two gigabytes.

Up to 255 fields per record.

Record size: Up to 10,800 bytes per record for indexed tables and 32,750 bytes per record for nonindexed tables. When figuring out the size (the number of bytes or characters) of a table, remember that Alpha fields take up their size (for example, an

A10 = 10 bytes), numeric field types take up 8 bytes, short number field types take up 2 bytes, money takes up 8, and dates take up 4 bytes.

Memos, BLOBs, and so on take 10 bytes plus however much of the memo is stored in the .DB. For example, M15 takes 25 bytes.

Up to 64 validity checks per table for Paradox for Windows tables.

A primary index can have up to 16 fields.

Tables can have up to 127 secondary indexes.

Block size can be from 1K to 32K in steps of 1K. For example, 1024, 2048, 3072, 4096, 5120...32768.

1.2.6 Paradox 7 and above Table Specifications

The Paradox 7 table format was introduced in Paradox version 7 for Windows 95/NT. The Paradox 7 table format has all the same specifications as the Paradox 5 table format with two additions. Following are the specification additions for the Paradox 7 table format.

- Added descending secondary indexes.
- Added unique secondary indexes

1.2.7 Paradox Field Types

Alpha (A)

Paradox 3.5, 4, 5, and 7 field type that can contain up to 255 letters and numbers. This field type was called Alphanumeric in versions of Paradox before version 5. It is similar to the Character field type in dBASE.

Autoincrement (+)

Field type introduced in the Paradox 5 table format that adds one to the highest number in the table whenever a record is inserted. The starting range can from -2,147,483,647 to 2,147,483,647. Deleting a record does not change the field values of other records.

BCD (#)

Paradox 5 and 7 field type which is provided only for compatibility with other applications that use BCD data. Paradox correctly interprets BCD data from other applications that use the BCD type. When Paradox performs calculations on BCD data, it converts the data to the numeric float type, then converts the result back to BCD. When this field type is fully supported, it will support up to 32 significant digits.

Binary (B)

Paradox 1, 5, and 7 field type that can store binary data up to 256MB per field.

Bytes (Y)

Paradox 5 and 7 field type for storing binary data up to 255 bytes. Unlike binary fields, bytes fields are stored in the Paradox table (rather than in the separate .MB file), allowing for faster access.

Date (D)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V. dBASE tables can store dates from January 1, 100, to December 31, 9999. Paradox 5 tables can store from 12/31/9999 B.C. to 12/31/9999 A.D.

Formatted Memo (F)

Paradox 1, 4.5, 5, and 7 field type is like a memo field except that you can format the text. You can alter and store the text attributes of typeface, style, color, and size. This rich text document has a variable-length up to 256MB per field.

Graphic (G)

Paradox 1, 5, and 7 field type can contain pictures in .BMP (up to 24 bit), .TIF (up to 256 color), .GIF (up to 256 color), .PCX, and .EPS file formats. Not all graphic variations are available. For example, currently you cannot store a 24-bit .TIF graphic. When you paste a graphic into a graphic field, Paradox converts the graphic into the .BMP format.

Logical (L)

Paradox 5 and 7 and dBASE III+, IV, and V field type can store values representing True or False (yes or no). By default, valid entries include T and F (case is not important).

Memo (M)

Paradox 4, 5, and 7 as well as dBASE III+, IV, and V field. A Paradox field type is an Alpha variable-length field up to 256MB per field. dBASE Memo fields can contain binary as well as memo data.

For Paradox tables, the file is divided into blocks of 512 characters. Each block is referenced by a sequential number, beginning at zero. Block 0 begins with a 4-byte number in hexadecimal format, in which the least significant byte comes first. This number specifies the number of the next available block. It is, in effect, a pointer to the end of the memo file. The remainder of Block 0 isn't used.

Money ($)

Paradox 3.5, 4, 5, and 7 field type, like number fields, can contain only numbers. They can hold positive or negative values. Paradox recognizes up to six decimal places when performing internal calculations on money fields. This field type was called Currency in previous versions of Paradox.

OLE (O)

Paradox 1, 5, and 7 as well as dBASE V field type that can store OLE data.

Number (N)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V field type can store up to 15 significant digits -10307 to + 10308 with up to 15 significant digits.

dBASE number fields contain numeric data in a Binary Coded Decimal (BCD) format. Use number fields when you need to perform precise calculations on the field data. Calculations on number fields are performed more slowly but with greater precision than are calculations on float number fields. The size of a dBASE number field can be from 1 to 20. Remember, however, that BCD is in Paradox 5 and 7 only for compatibility and is mapped directly to the Number field type.

Short (S)

Paradox 3.5, 4, 5, and 7 field type that can contain integers from -- 32,767 through 32,767 (no decimal).

Time (T)

Paradox 5 and 7 field type that can contain time times of day, stored in milliseconds since midnight and limited to 24 hours.

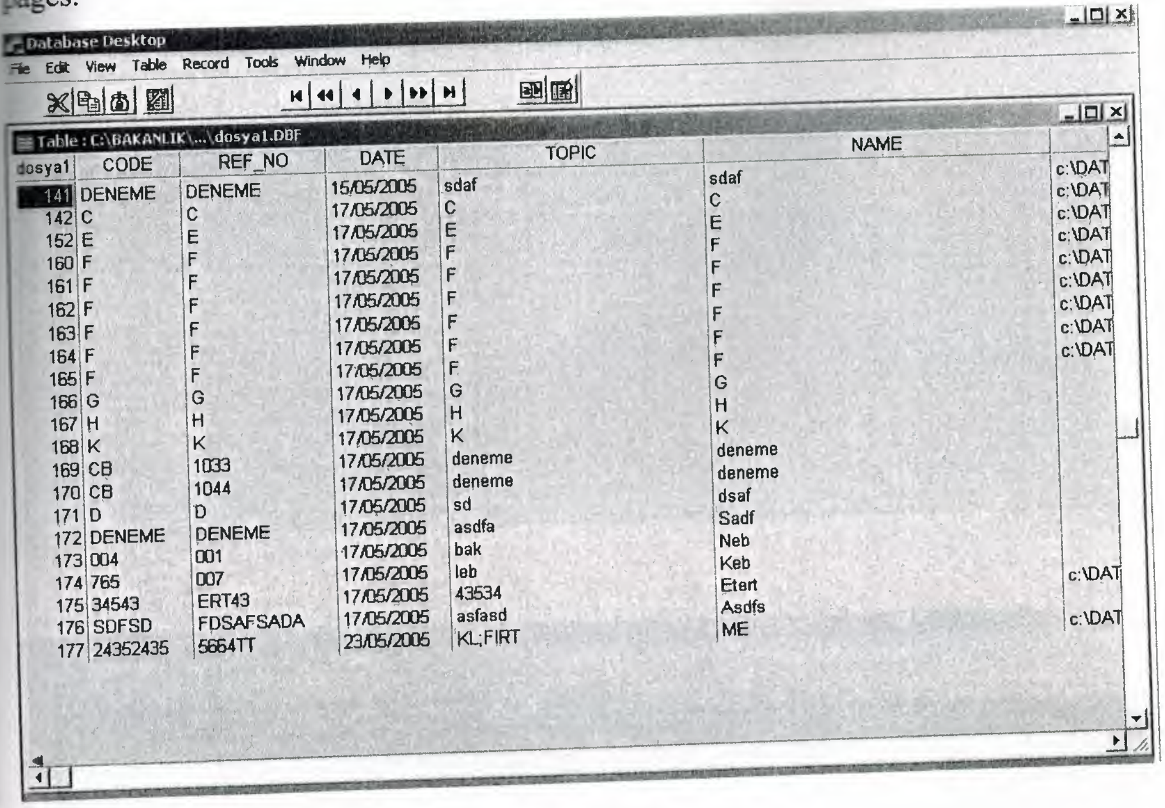
This field type does not store duration which is the difference between two times. For example, if you need to store the duration of a song, use an Alpha field. Whenever you need to store time, make a distinction between clock time and duration. The Time field type is perfect for clock time. Duration can be stored in an Alpha field and manipulated with code.

TimeStamp (@)

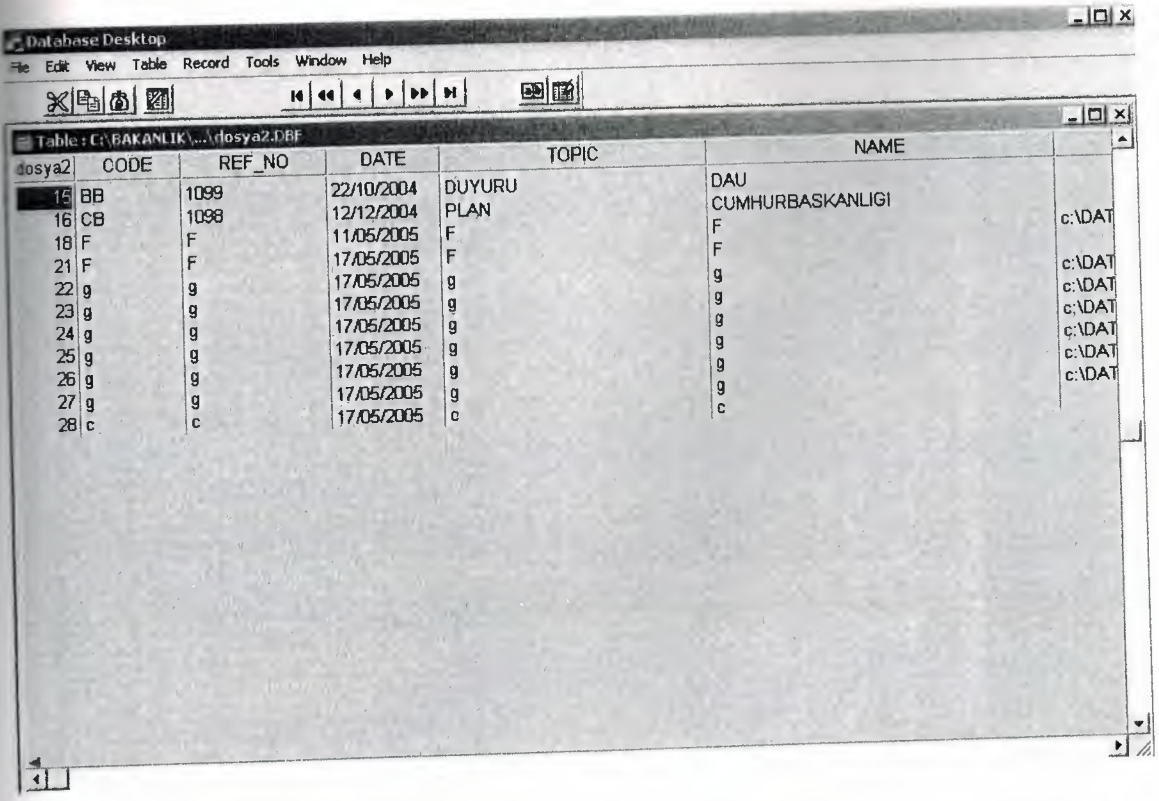
Paradox 5 field type comprised of both date and time values. Rules for this field type are the same as those for date fields and time fields.

1.2.8 About Our Database

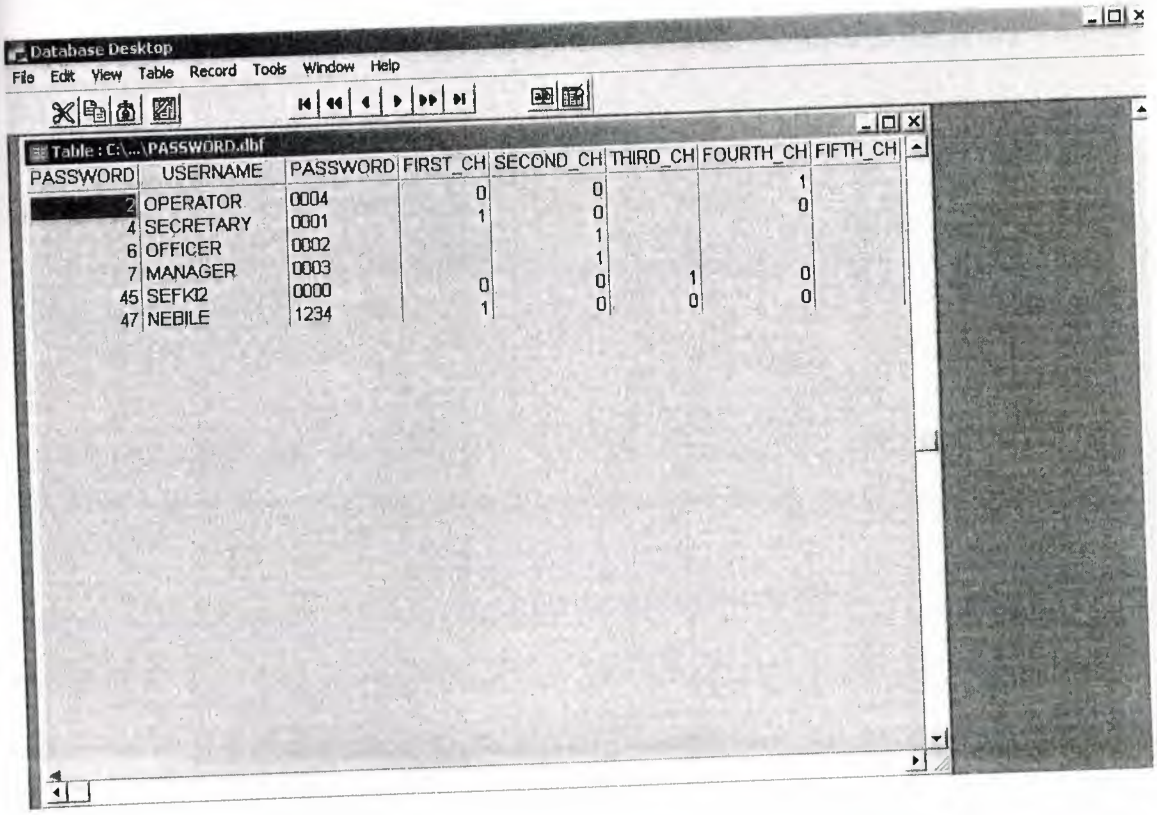
In our archive program we used dbase for database .You can see the tables in following pages.


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\BAKANLIK\...\dosya1.DBF

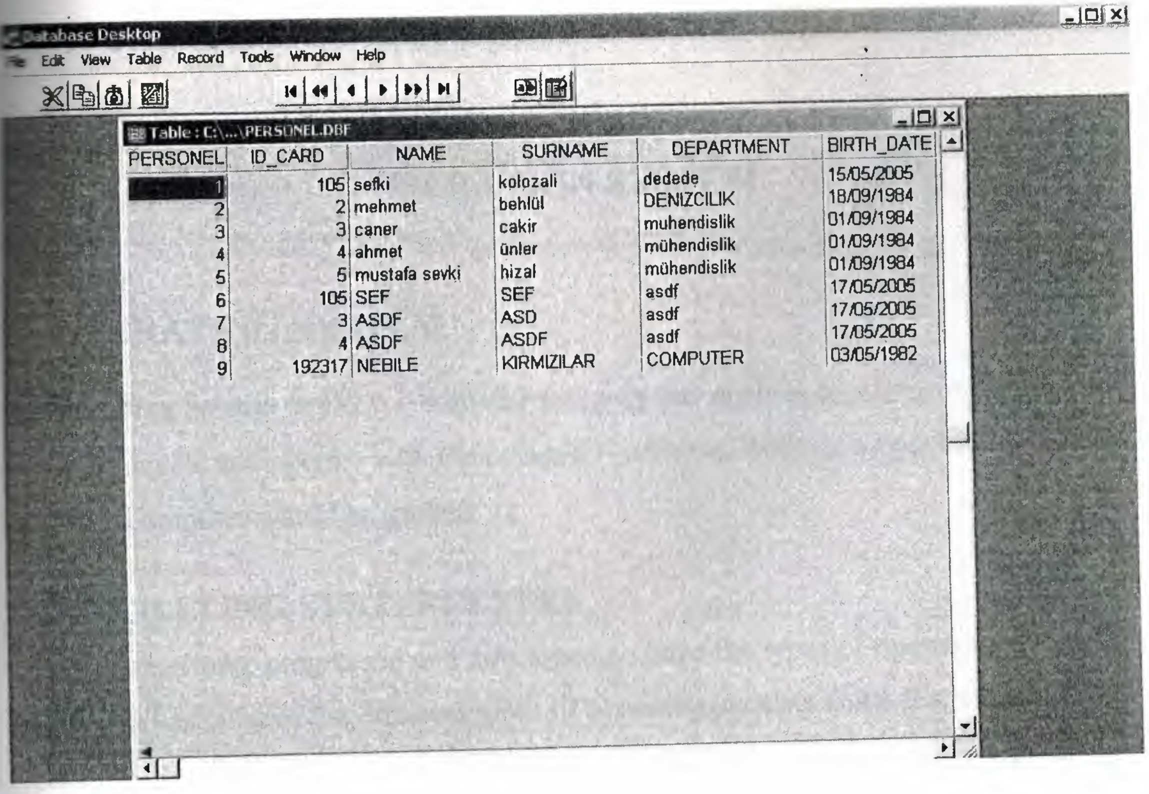
dosya1	CODE	REF_NO	DATE	TOPIC	NAME	
141	DENEME	DENEME	15/05/2005	sdaf	sdaf	c:\DAT
142	C	C	17/05/2005	C	C	c:\DAT
152	E	E	17/05/2005	E	E	c:\DAT
160	F	F	17/05/2005	F	F	c:\DAT
161	F	F	17/05/2005	F	F	c:\DAT
162	F	F	17/05/2005	F	F	c:\DAT
163	F	F	17/05/2005	F	F	c:\DAT
164	F	F	17/05/2005	F	F	c:\DAT
165	F	F	17/05/2005	F	F	c:\DAT
166	G	G	17/05/2005	G	G	
167	H	H	17/05/2005	H	H	
168	K	K	17/05/2005	K	K	
169	CB	1033	17/05/2005	deneme	deneme	
170	CB	1044	17/05/2005	deneme	deneme	
171	D	D	17/05/2005	sd	dsaf	
172	DENEME	DENEME	17/05/2005	asdfa	Sadf	
173	004	001	17/05/2005	bak	Neb	
174	765	007	17/05/2005	leb	Keb	
175	34543	ERT43	17/05/2005	43534	Etert	c:\DAT
176	SDFSD	FDSAFSADA	17/05/2005	asfasd	Asdfs	
177	24352435	5664TT	23/05/2005	KL;FIRT	ME	c:\DAT


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\BAKANLIK\...\dosya2.DBF

dosya2	CODE	REF_NO	DATE	TOPIC	NAME	
15	BB	1099	22/10/2004	DUYURU	DAU	
16	CB	1098	12/12/2004	PLAN	CUMHURBASKANLIGI	
18	F	F	11/05/2005	F	F	c:\DAT
21	F	F	17/05/2005	F	F	
22	g	g	17/05/2005	g	g	c:\DAT
23	g	g	17/05/2005	g	g	c:\DAT
24	g	g	17/05/2005	g	g	c:\DAT
25	g	g	17/05/2005	g	g	c:\DAT
26	g	g	17/05/2005	g	g	c:\DAT
27	g	g	17/05/2005	g	g	c:\DAT
28	c	c	17/05/2005	c	c	


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\...\PASSWORD.dbf

PASSWORD	USERNAME	PASSWORD	FIRST_CH	SECOND_CH	THIRD_CH	FOURTH_CH	FIFTH_CH
2	OPERATOR	0004	0	0		1	
4	SECRETARY	0001	1	0		0	
6	OFFICER	0002		1			
7	MANAGER	0003		1			
45	SEFKI2	0000	0	0	1	0	
47	NEBILE	1234	1	0	0	0	


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\...\PERSONEL.DBF

PERSONEL	ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
1	105	sefki	kolozali	dedede	15/05/2005
2	2	mehmet	behlül	DENIZCILIK	18/09/1984
3	3	caner	cakir	muhendislik	01/09/1984
4	4	ahmet	ünler	mühendislik	01/09/1984
5	5	mustafa sevki	hizal	mühendislik	01/09/1984
6	105	SEF	SEF	asdf	17/05/2005
7	3	ASDF	ASD	asdf	17/05/2005
8	4	ASDF	ASDF	asdf	17/05/2005
9	192317	NEBILE	KIRMIZILAR	COMPUTER	03/05/1982

CHAPTER 2

OPERATION SYSTEM

2.1 OPERATING SYSTEM

An Operating System or OS is a software program that enables the computer hardware to communicate and operate with the computer software. Without a computer Operating System a computer would be useless.

2.2 OPERATING SYSTEM TYPES

As computers have progressed and developed so have the types of operating systems. Below is a basic list of the different types of operating systems and a few examples of Operating Systems that fall into each of the categories. Many computer Operating Systems will fall into more then one of the below categories.

2.2.1 Multi-user

A multi-user Operating System allows for multiple users to use the same computer at the same time and/or different times.

Below are some examples of multi-user Operating Systems.

Linux
UNIX
Windows 2000

2.2.2 Multiprocessing

An Operating System capable of supporting and utilizing more than one computer processor.

1. When referring to a network, a multi-user system is a term commonly used to define a computer capable of allowing multiple users to connect to a network.

2. When referring to a computer Operating System, a multi-user system is a computer with an Operating system that supports multiple users at once and/or different times.

Below are some examples of multiprocessing Operating Systems.

Linux
UNIX
Windows 2000

2.2.3 Multitasking

An Operating systems that is capable of allowing multiple software processes to be run at the same time. Below are some examples of multitasking Operating Systems.

UNIX
Windows 2000

2.2.4 Multithreading

Operating systems that allow different parts of a software program to run concurrently. Operating systems that would fall into this category are:

Linux
UNIX
Windows 2000

2.3 Introduction to UNIX:

2.3.1 Objectives

Its covers:

- The concept of an operating system.
- The internal architecture of an operating system.
- The evolution of the UNIX operating system into two broad schools (BSD and SYSV) and the development of Linux, a popular open source operating system.

- The architecture of the Linux operating system in more detail.
- How to log into (and out of) UNIX and change your password.
- The general format of UNIX commands.

2.3.2 What is an Operating System?

An operating system (OS) is a resource manager. It takes the form of a set of software routines that allow users and application programs to access system resources (e.g. the CPU, memory, disks, modems, printers network cards etc.) in a safe, efficient and abstract way.

For example, an OS ensures safe access to a printer by allowing only one application program to send data directly to the printer at any one time. An OS encourages efficient use of the CPU by suspending programs that are waiting for I/O operations to complete to make way for programs that can use the CPU more productively. An OS also provides convenient abstractions (such as files rather than disk locations) which isolate application programmers and users from the details of the underlying hardware.

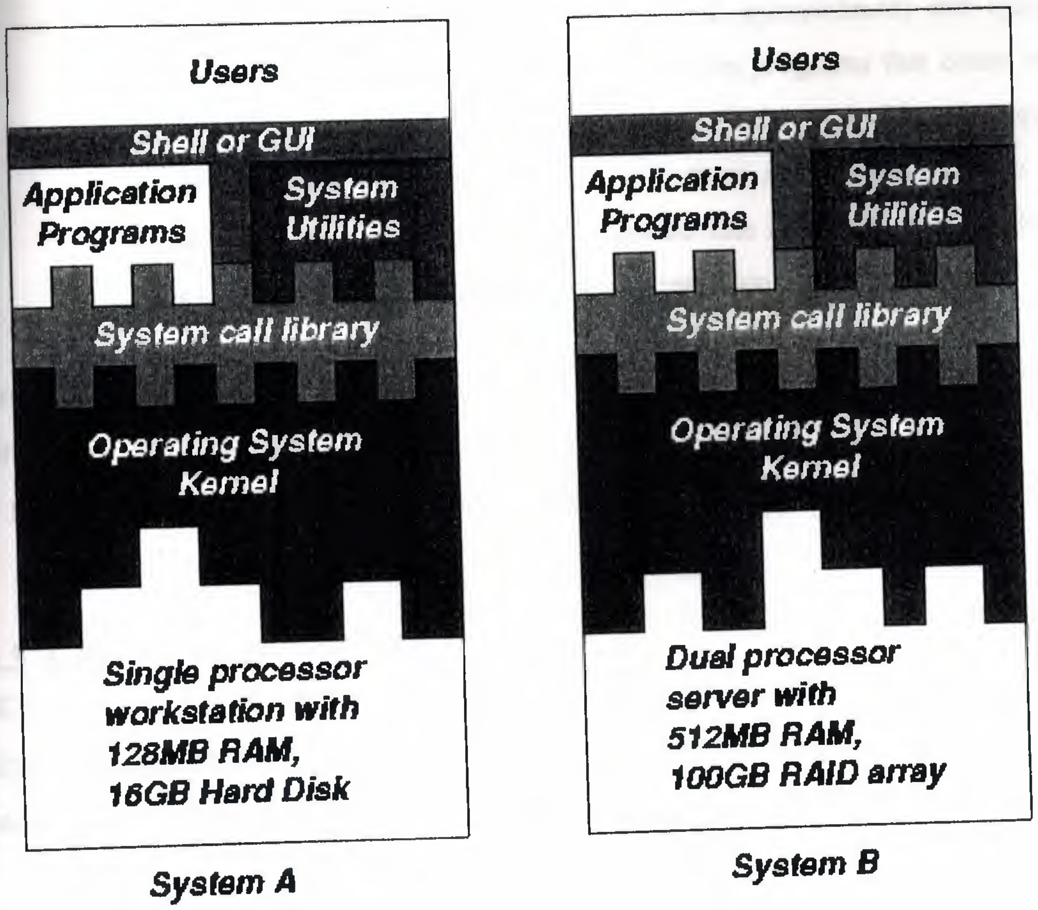


Fig. 1.1: General operating system architecture

Fig. 1.1 presents the architecture of a typical operating system and shows how an OS succeeds in presenting users and application programs with a uniform interface without regard to the details of the underlying hardware. We see that:

- The operating system kernel is in direct control of the underlying hardware. The kernel provides low-level device, memory and processor management functions (e.g. dealing with interrupts from hardware devices, sharing the processor among multiple programs, allocating memory for programs etc.)
- Basic hardware-independent kernel services are exposed to higher-level programs through a library of system calls (e.g. services to create a file, begin execution of a program, or open a logical network connection to another computer).

- Application programs (e.g. word processors, spreadsheets) and system utility programs (simple but useful application programs that come with the operating system, e.g. programs which find text inside a group of files) make use of system calls. Applications and system utilities are launched using a shell (a textual command line interface) or a graphical user interface that provides direct user interaction.

Operating systems (and different flavours of the same operating system) can be distinguished from one another by the system calls, system utilities and user interface they provide, as well as by the resource scheduling policies implemented by the kernel.

2.3.3 A Brief History of UNIX

UNIX has been a popular OS for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities. What follows here is a simplified history of how UNIX has developed.

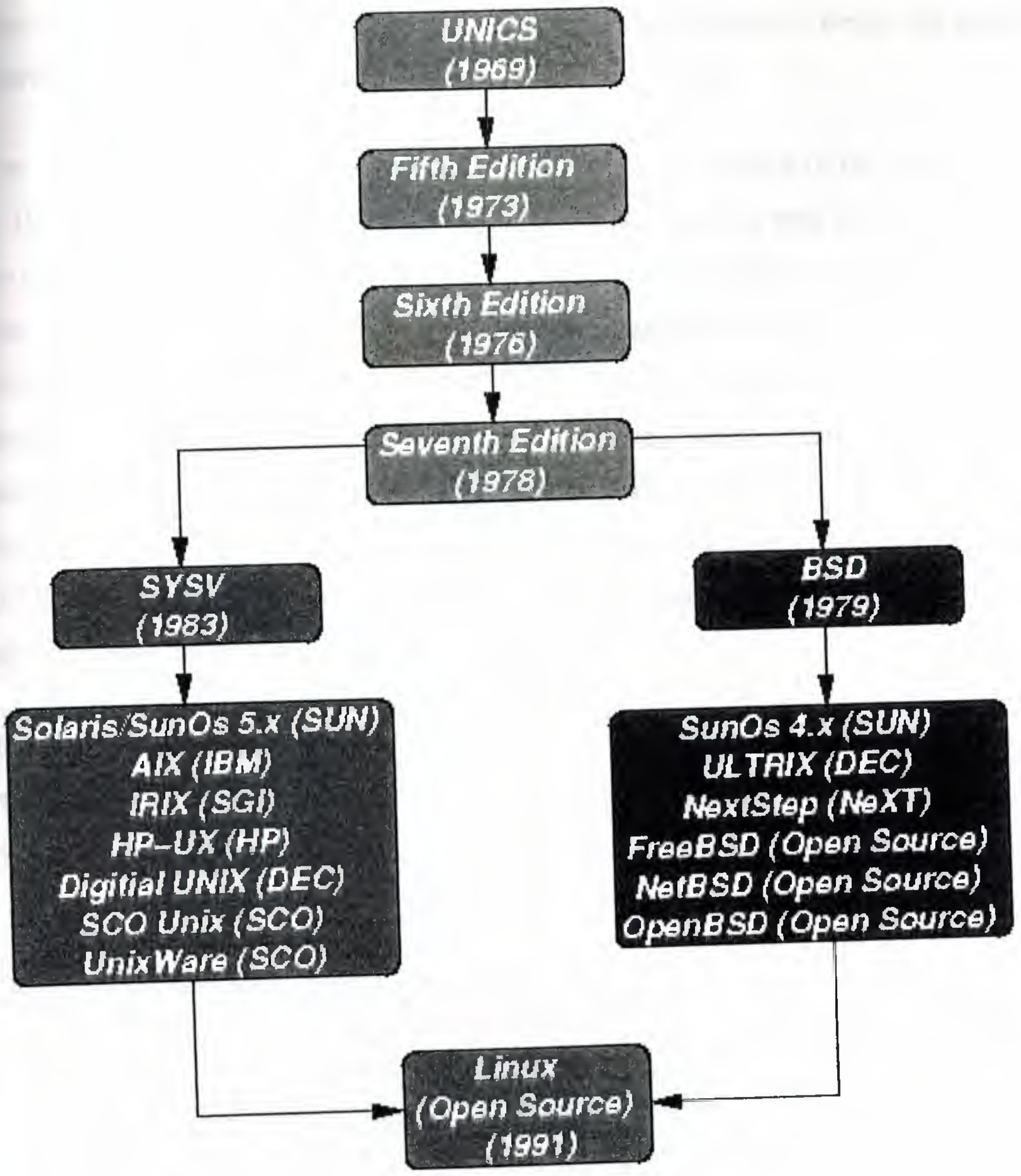


Fig. 1.2: Simplified UNIX FamilyTree

In the late 1960s, researchers from General Electric, MIT and Bell Labs launched a joint project to develop an ambitious multi-user, multi-tasking OS for mainframe computers known as MULTICS (Multiplexed Information and Computing System). MULTICS failed (for some MULTICS enthusiasts "failed" is perhaps too strong a word to use here), but it did inspire Ken Thompson, who was a researcher at Bell Labs, to have a go at writing a simpler operating system himself. He wrote a simpler version of MULTICS on a PDP7 in assembler and called his attempt UNICS (Uniplexed Information and Computing System). Because memory and CPU power were at a premium in those days, UNICS (eventually shortened to UNIX) used short commands to minimize the

space needed to store them and the time needed to decode them - hence the tradition of short UNIX commands we use today, e.g. ls, cp, rm, mv etc.

Ken Thompson then teamed up with Dennis Ritchie, the author of the first C compiler in 1973. They rewrote the UNIX kernel in C - this was a big step forwards in terms of the system's portability - and released the Fifth Edition of UNIX to universities in 1974. The Seventh Edition, released in 1978, marked a split in UNIX development into two main branches: SYSV (System 5) and BSD (Berkeley Software Distribution). BSD arose from the University of California at Berkeley where Ken Thompson spent a sabbatical year. Its development was continued by students at Berkeley and other research institutions. SYSV was developed by AT&T and other commercial companies. UNIX flavours based on SYSV have traditionally been more conservative, but better supported than BSD-based flavours.

The latest incarnations of SYSV (SVR4 or System 5 Release 4) and BSD Unix are actually very similar. Some minor differences are to be found in file system structure, system utility names and options and system call libraries as shown in Fig 1.3.

Feature	**Typical SYSV**	**Typical BSD**
kernel name	/unix	/vmunix
boot init	/etc/rc.d directories	/etc/rc.* files
mounted FS	/etc/mnttab	/etc/mtab
default shell	sh, ksh	csh, tcsh
FS block size	512 bytes->2K	4K->8K
print subsystem	lp, lpstat, cancel	lpr, lpq, lprm
echo command (no new line)	echo "\c"	echo -n
ps command	ps -fae	ps -aux
multiple wait syscalls	poll	select
memory access syscalls	memset, memcpy	bzero, bcopy

Fig. 1.3: Differences between SYSV and BSD

Linux is a free open source UNIX OS for PCs that was originally developed in 1991 by Linus Torvalds, a Finnish undergraduate student. Linux is neither pure SYSV or pure BSD. Instead, incorporates some features from each (e.g. SYSV-style startup files but BSD-style file system layout) and aims to conform with a set of IEEE standards called POSIX (Portable Operating System Interface). To maximise code portability, it typically supports SYSV, BSD and POSIX system calls (e.g. poll, select, memset, memcpy, bzero and bcopy are all supported).

The open source nature of Linux means that the source code for the Linux kernel is freely available so that anyone can add features and correct deficiencies. This approach has been very successful and what started as one person's project has now turned into a collaboration of hundreds of volunteer developers from around the globe. The open source approach has not just successfully been applied to kernel code, but also to application programs for Linux.

As Linux has become more popular, several different development streams or distributions have emerged, e.g. Redhat, Slackware, Mandrake, Debian, and Caldera. A distribution comprises a prepackaged kernel, system utilities, GUI interfaces and application programs.

Redhat is the most popular distribution because it has been ported to a large number of hardware platforms (including Intel, Alpha, and SPARC), it is easy to use and install and it comes with a comprehensive set of utilities and applications including the X Windows graphics system, GNOME and KDE GUI environments, and the StarOffice suite (an open source MS-Office clone for Linux).

2.3.4 Architecture of the Linux Operating System

Linux has all of the components of a typical OS (at this point you might like to refer back to Fig 1.1):

- **Kernel**

  The Linux kernel includes device driver support for a large number of PC hardware devices (graphics cards, network cards, hard disks etc.), advanced processor and memory management features, and support for many different types of filesystems (including DOS floppies and the

ISO9660 standard for CDROMs). In terms of the services that it provides to application programs and system utilities, the kernel implements most BSD and SYSV system calls, as well as the system calls described in the POSIX.1 specification.

The kernel (in raw binary form that is loaded directly into memory at system startup time) is typically found in the file /boot/vmlinuz, while the source files can usually be found in /usr/src/linux.The latest version of the Linux kernel sources can be downloaded from http://www.kernel.org.

- **Shells and GUIs**
  Linux supports two forms of command input: through textual command line shells similar to those found on most UNIX systems (e.g. sh - the Bourne shell, bash - the Bourne again shell and csh - the C shell) and through graphical interfaces (GUIs) such as the KDE and GNOME window managers. If you are connecting remotely to a server your access will typically be through a command line shell.

- **System Utilities**
  Virtually every system utility that you would expect to find on standard implementations of UNIX (including every system utility described in the POSIX.2 specification) has been ported to Linux. This includes commands such as ls, cp, grep, awk, sed, bc, wc, more, and so on. These system utilities are designed to be powerful tools that do a single task extremely well (e.g. grep finds text inside files while wc counts the number of words, lines and bytes inside a file). Users can often solve problems by interconnecting these tools instead of writing a large monolithic application program.

Like other UNIX flavours, Linux's system utilities also include server programs called daemons which provide remote network and administration services (e.g. telnetd and sshd provide remote login facilities, lpd provides printing services, httpd serves web

pages, crond runs regular system administration tasks automatically). A daemon (probably derived from the Latin word which refers to a beneficient spirit who watches over someone, or perhaps short for "Disk And Execution MONitor") is usually spawned automatically at system startup and spends most of its time lying dormant (lurking?) waiting for some event to occur.

- **Application programs**

  Linux distributions typically come with several useful application programs as standard. Examples include the emacs editor, xv (an image viewer), gcc (a C compiler), g++ (a C++ compiler), xfig (a drawing package), latex (a powerful typesetting language) and soffice (StarOffice, which is an MS-Office style clone that can read and write Word, Excel and PowerPoint files).

  Redhat Linux also comes with rpm, the Redhat Package Manager which makes it easy to install and uninstall application programs.

2.4 An Introduction to Linux

2.4.1 What is Linux?

Linux is a UNIX-based operating system originally developed as for Intel-compatible PC's. It is now available for most types of hardware platforms, ranging from PDAs (and according to some reports, a wristwatch) to mainframes. Linux is a "modern operating system", meaning it has such features as virtual memory, memory protection, and preemptive multitasking.

Linux is built and supported by a large international community of developers and users dedicated to free, open-source software. This community sees Linux as an alternative to such proprietary systems as Windows and Solaris, and as a platform for alternatives to such proprietary applications as MS Office, Internet Explorer, and Outlook.

As a result of this community, there is a very large collection of free software available for Linux. There are graphical environments (GUIs), office applications, developers'

tools, system utilities, business applications, document publishing tools, network client and server applications -- the list goes on.

The best part of this community is that all code is open. This means there is no barrier to entry; for any given problem, there are generally several applications that solve the problem. These applications can also borrow the best parts from each other to become even better. An excellent example of this is Galeon. Galeon is a web browser which took Mozilla's web page rendering engine and integrated it with a GTK frontend (instead of Mozilla's normal frontend).

Linux specifically refers to the Linux kernel. However, the kernel is useless without a set of tools and applications to run on the kernel. Linux is most commonly distributed with this toolset and a collection of applications in what is called a "distribution". The most common are Redhat, Mandrake, Suse, and Debian. Distributions differ in three basic ways: the process for installing the distribution, the applications available, and process for installing and managing these applications.

2.4.2 Why use Linux?

Reasons to Install Linux

- Configurability
- Convenience
- Stability
- Community
- Freedom

2.4.2.1 Configurability

Linux distributions give the user full access to configure just about any aspect of their system. Options range from the simple and straightforward (for instance, changing the background image) to the more esoteric (for instance, making the "Caps Lock" key behave like "Control"). Almost any aspect of the user experience can be configured.

Linux also allows the user to automate just about any task. Advanced scripting and high-level programming are standard features. Most operations are accessible via these scripting options. Finally, Linux offers the ultimate in configurability: the source code, to be modified as you see fit.

2.4.2.2 Convenience

While Linux takes some effort to get set up, once it is set up, it is surprisingly low-maintenance. Package management can simply be a matter of running two commands in the shell. Linux also offers complete remote access. This allows the user to act exactly as if she is sitting at that computer's desk, potentially across town or on the other side of the world.

2.4.2.3 Stability

Linux is based on the UNIX kernel. It provides preemptive multitasking and protected memory. Preemptive multitasking prevents any application from permanently stealing the CPU and locking up the machine. Protected memory prevents applications from interfering with and crashing one-another.

Linux and related tools are also open-source. This means that the source code is available for the public to view. There are literally hundreds, if not thousands, of developers working on the various pieces of Linux. In this open development process, bugs are fixed very quickly. In addition, bugs are fixed immediately, instead of waiting for the next major release. It certainly helps that the people who develop Linux and associated tools use their programs every day.

2.4.2.4 Community

Linux is part of the greater open-source community. This consists of thousands of developers and many more users world-wide who support open software. This user and developer base is also a support base.

In Rice, there is the Rice Linux Users Group (the group who are bringing you this class). We hold regular meetings where people can bring up their Linux problems. There is also the newsgroup rice.comp.linux, where questions can be asked or problems laid out any time, day or night. This newsgroup is mirrored to the RLUG-discuss mailing list, for RLUG members who don't have access to Rice newsgroups.

Worldwide, the Linux community is even greater. There is a mailing list for just about every project or piece of software in active development -- if you have a question about

a program, who better to ask than the person who wrote it? There are also newsgroups and web pages which have collectively with the mailing lists probably addressed every problem someone new to Linux has encountered several times over.

2.4.2.5 Freedom

Linux is free. This means more than just costing nothing. This means that you are allowed to do whatever you want to with the software. This is why Redhat, Mandrake, and Suse are all allowed to sell their own distributions of Linux. The only restriction placed on Linux is that, if you distribute Linux, you must grant all the privileges to the code that you had, including providing the source. This prevents a corporation from using the Linux kernel as the basis for their proprietary operating system.

2.5 The differences between Linux and other UNIX

The differences between Linux and other UNIX operating systems is barely noticable to the average user. Commands might be located in slightly different places or may have different sets of arguments but the functionallity is the same. With the GNU tools available for most platforms these days even these differences are starting to vanish. Most of the differences are located in the kernel design and the systems administration tools. Some systems such as AIX and HP-UX provide fancy graphical management programs while others such as Linux and Solaris tend to rely on editing files and running commands manually. To sum up, UNIX is like a car: once you can drive one, you can drive them all. However if you open up the bonnet to tinker around you'll find many diverse systems as you move from one type to another.

CHAPTER 3

DELPHI

3.1.INTRODUCTION

The name "Delphi" was never a term with which either Olaf Helmer or Norman Dalkey (the founders of the method) were particular happy. Since many of the early Delphi studies focused on utilizing the technique to make forecasts of future occurrences, the name was first applied by some others at Rand as a joke. However, the name stuck. The resulting image of a priestess, sitting on a stool over a crack in the earth, inhaling sulfur fumes, and making vague and jumbled statements that could be interpreted in many different ways, did not exactly inspire confidence in the method.

The straightforward nature of utilizing an iterative survey to gather information "sounds" so easy to do that many people have done "one" Delphi, but never a second. Since the name gives no obvious insight into the method and since the number of unsuccessful Delphi studies probably exceeds the successful ones, there has been a long history of diverse definitions and opinions about the method. Some of these misconceptions are expressed in statements such as the following that one finds in the literature:

It is a method for predicting future events.

It is a method for generating a quick consensus by a group.

It is the use of a survey to collect information.

It is the use of anonymity on the part of the participants.

It is the use of voting to reduce the need for long discussions.

It is a method for quantifying human judgement in a group setting.

Some of these statements are sometimes true; a few (e.g. consensus) are actually contrary to the purpose of a Delphi. Delphi is a communication structure aimed at

producing detailed critical examination and discussion, not at forcing a quick compromise. Certainly quantification is a property, but only to serve the goal of quickly identifying agreement and disagreement in order to focus attention. It is often very common, even today, for people to come to a view of the Delphi method that reflects a particular application with which they are familiar. In 1975 Linstone and Turoff proposed a view of the Delphi method that they felt best summarized both the technique and its objective:

"Delphi may be characterized as a method for structuring a group communication process, so that the process is effective in allowing a group of individuals, as a whole, to deal with complex problems." (page 3)

The essence of Delphi is structuring of the group communication process. Given that there had been much earlier work on how to facilitate and structure face-to-face meetings, the other important distinction was that Delphi was commonly applied utilizing a paper and pencil communication process among groups in which the members were dispersed in space and time. Also, Delphis were commonly applied to groups of a size (30 to 100 individuals) that could not function well in a face-to-face environment, even if they could find a time when they all could get together.

Additional opportunity has been added by the introduction of Computer Mediated Communication Systems (Hiltz and Turoff, 1978; Rice and Associates, 1984; Turoff, 1989; Turoff, 1991). These are computer systems that support group communications in either a synchronous (Group Decision Support Systems, Desanctis et. al., 1987) or an asynchronous manner (Computer Conferencing). Techniques that were developed and refined in the evolution of the Delphi Method (e.g. anonymity, voting) have been incorporated as basic facilities or tools in many of these computer based systems. As a result, any of these systems can be used to carry out some form of a Delphi process or Nominal Group Technique (Delbecq, et. al., 1975).

The result, however, is not merely confusion due to different names to describe the same things; but a basic lack of knowledge by many people working in these areas as to what was learned in the studies of the Delphi Method about how to properly employ these techniques and their impact on the communication process. There seems to be a great deal of "rediscovery" and repeating of earlier misconceptions and difficulties.

Given this situation, the primary objective of this chapter is to review the specific properties and methods employed in the design and execution of Delphi Exercises and to examine how they may best be translated into a computer based environment.

3.2. ASYNCHRONOUS INTERACTION

Perhaps the most important and least understood property of the Delphi method is the ability of members of a group to participate in an asynchronous manner. This property of asynchronous interaction has two characteristics:

A person may choose to participate in the group communication process when they feel they want to.

A person may choose to contribute to that aspect of the problem to which they feel best able to contribute.

It does not matter what time of the day or night Delphi participants think of good ideas to include in their response. They can fill out a Delphi survey when they wish to, or they can go to a computer terminal to contribute when they wish to. This can be done at whatever point in time the individual feels he or she has thought of significant things to include in response to the issues involved. Participants can revise and add to their responses over time, before sending them to the group monitor for dissemination to the others.

A good Delphi survey attempts to tackle the problem from many different perspectives. Sometimes this is referred to as including questions in the Delphi survey which approach the problem both from the "bottom-up" and from the "top-down" perspectives. This allows different individuals in the group to focus on the approach to problem solving with which they feel most comfortable.

In a normal face-to-face group process, and in the environment characterized by face-to-face Group Decision Support Systems, all the members of the group are forced into a lockstep treatment of a problem. When the group is considering the subject of "goals," those who have difficulty dealing with "abstraction" may feel at a disadvantage, because they do not have as much to contribute. Conversely, when focusing on specific solution approaches, those who deal better with "abstraction" may not feel they are contributing. One of the specific advantages of groups is to allow individuals with differing perspectives and/or differing cognitive abilities to contribute to those parts of a complex

problem for which they have both the appropriate knowledge and appropriate problem solving skills. A typical model for a group problem solving process is:
Recognition of the problem
Defining the problem
Changing the representation of the problem
Developing the goals associated with solving the problem
Determining the strategy for generating the possible solutions
Choosing a strategy
Generating the evaluation criteria to be applied to solutions
Evaluating the solution criteria
Generating the solutions
Evaluating the solutions
The literature on cognitive abilities and human problem solving does confirm that individuals differ considerably, based upon their cognitive abilities (Benbasat and Taylor, 1982; Streitz, 1987), in their ability to deal with different aspects of a problem solving situation. This depends upon such psychological dimensions as their ability to deal with Abstraction - No Abstraction, Search - No Search, Data Driven - Conceptually Driven, and Deductive - Inductive cognitive processes.
In most face-to-face approaches, the group is forced as a whole to take a sequential path through a group problem solving process. In the Delphi process, we try to design a communication structure that allows any individual to choose the sequence in which to examine and contribute to the problem solving process. This is the single most important criterion by which we should evaluate the design of a Delphi oriented communication structure. Does it allow the individual to exercise personal judgement about what part of the problem to deal with at any time in the group problem solving process?
It is actually easier to accomplish this using a computer system than it has been with paper and pencil based Delphi studies. The "round" structure and the need to limit the physical size of any paper and pencil survey places severe constraints on the degree to which one can carry out the above approach. Hence, paper and pencil Delphis are usually limited by the "top-down/bottom-up" dichotomy rather than allowing more complete parallel entry to any aspect of the problem. For example, in a single Delphi one might explore on the first round "goals" (a top view) and specific "consequences" (a

bottom view). Relating goals to consequences requires developing the relationships inherent in alternative actions and states of nature. These would be put off to a later round. In the computerized environment individuals could be free to tackle any aspect of the problem according to personal preferences.

This particular objective of Delphi design is also characterized by two other practices commonly applied to Delphi studies. First, it should be clear to the respondents that they do not have to respond to every question, but can decide to take a "no judgement" view. Secondly, one usually solicits the respondent's confidence in their judgements, particularly when they are quantified judgements. This has been found to improve the quality of the estimates made in Delphi exercises (Dalkey, 1970). This allows the respondents to estimate their own degree of expertise on the judgements they are supplying. The fact that contributions can be made anonymously also means a person does not have to feel embarrassment if he or she does not feel able to confidently contribute to a specific aspect of the problem.

This advantage for the Delphi approach comes at an obvious price. With material being supplied in parallel, it is clear that the need to structure and organize it in a manner that it makes sense to the group is a primary requirement (Turoff, 1974, 1991; Hiltz and Turoff, 1985). The need to carefully define the total communication structure and put it into a framework that produces both a group view and a synchronization of the group process is the most difficult part of a good Delphi design. We will treat this in following sections. In paper and pencil Delphis, this is the effort that must be undertaken by the design team in processing the results of each round and producing a proper summary. In a computer based Delphi process, this has a somewhat different connotation in that the round structure disappears, replaced by a continuous feedback process which may or may not involve human intervention for the processing.

The most significant observation resulting from the above considerations, is that most of the attempts to understand the group problem solving process in the computer based environment are still based upon models that were developed from studying face-to-face groups. Thus, what are often thought of as being "ideal" group problem solving structures are based upon the "sequential" treatment of a problem by a group (Turoff, 1991). There has been little work to date to develop models of the group problem solving process that are based upon parallel and asynchronous activities by the individuals within the group. There is need for a model which integrates the individual

problem solving process with the group process. It is only within the context of such a model that we can come to a deeper understanding of the design process that goes beyond the trial and error evolution of the method that has occurred to date.

3.3 ANONYMITY

Perhaps the property that most characterizes the Delphi method in the mind of most people is the use of anonymity. Typically, in paper and pencil Delphis there is no identification of who contributed specific material or who made a particular evaluative judgment about it. This property is not one that should be considered a hard and fast rule for all aspects of a Delphi. Moreover, the computer makes possible variations in anonymity not possible in a paper and pencil environment. Before we explore these, we should look at the primary reasons for anonymity:

Individuals should not have to commit themselves to initial expressions of an idea that may not turn out to be suitable.

If an idea turns out to be unsuitable, no one loses face from having been the individual to introduce it.

Persons of high status are reluctant to produce questionable ideas.

Committing one's name to a concept makes it harder to reject it or change one's mind about it.

Votes are more frequently changed when the identity of a given voter is not available to the group.

The consideration of an idea or concept may be biased by who introduced it.

When ideas are introduced within a group where severe conflicts exist in either "interests" or "values," the consideration of an idea may be biased by knowing it is produced by someone with whom the individual agrees or disagrees.

The high social status of an individual contributor may influence others in the group to accept the given concept or idea.

Conversely, lower status individuals may not introduce ideas, for fear that the idea will be rejected outright.

In essence, the objective of anonymity is to allow the introduction and evaluation of ideas and concepts by removing some of the common biases normally occurring in the face-to-face group process. Sometimes the use of anonymity has been carried too far.

For example, it is important that the members of a Delphi exercise believe that they are communicating with a peer group. An individual participant must feel that the other members of the group will be able to contribute valuable insight about the problem being examined. This is a primary factor in motivating participation. It is usual to inform the participants about who is actually involved in the group of Delphi respondents. Only when there are strong antagonisms among group members would one consider not doing this.

Delphi panelists are motivated to participate actively only if they feel they will obtain value from the information they receive as a result of the process. This value received needs to be at least equal, in their minds, to the effort expended to contribute information. This is one reason why blanket invitations to participate in a Delphi that do not specify who will be involved and what the feedback will be to the group members often result in very low participation rates.

When one introduces the concept of conducting a Delphi through a Computer Mediated Communication System, there are more options available for handling the process of anonymity. First, one can easily incorporate the use of pen-names (Hiltz, Turoff, and Johnson, 1989). While this does not identify who a person is, it does allow a person to be identified with a set of related contributions. This allows the other members of a group to obtain more understanding of why specific individuals are agreeing or disagreeing with certain concepts. For example, knowing all the arguments a person has made about accepting or rejecting a given position allows people to better tailor what needs to be said to perhaps change an individual's viewpoint. It also allows the expression of more complex individual viewpoints. This coherency is hard to observe or utilize when everything is anonymous.

As a result, it is probably desirable in most computer based Delphis to impose the default use of pen-names rather than anonymity on qualitative type statements made in the discussion. In some cases it is also possible to provide the privilege of allowing the respondents to choose when they wish to use pen-names and when they wish to use their real name. The more the individuals know one another and have a history as a "social" group, the more likely that good results will result from allowing participants to freely choose to use their real names, pen-names, or anonymity on qualitative type information.

There have been studies of computer based message systems which have attempted to conclude that the use of anonymity leads to "flaming" and antagonism (Kiesler, Siegel, and McGuire, 1984). Most of these observations have been based upon studying student groups who have no prior history or knowledge about one another. Flaming and disinhibition have not been problems among groups that already have a social history or social structure. When utilizing a computer based system with groups who are not familiar with one another, it may be important to provide a separate conference devoted to socializing among the group members. This would serve the same purpose as coffee breaks serve for co-located groups that work together. In the computer based communication environment, it has been observed that social-emotional exchanges are helpful in facilitating consensus development and eliminating potential misunderstanding (Hiltz, Johnson, and Turoff, 1986).

Anonymity for voting and estimates of subjective quantitative information is probably desirable to maintain in most circumstances. However, it is desirable that the coordinator for a Delphi exercise on the computer system be able to identify people with extreme votes or estimates. A Delphi coordinator should have no vested interest in the outcome and should be in a facilitation role. The facilitator may feel it is desirable to encourage individuals with extreme positions to explain them. Sometimes the observation that one is in a minority position can negatively affect participation unless there is such encouragement.

In some cases it may be desirable to allow voter identification. For example, in the final steps of a budget allocation task, it could be felt that everyone should assume final accountability for the recommended decision. Even in face-to-face committees, committee reports where no identified individuals assume responsibility have sometimes led to a lack of group commitment when it comes to implementing the results. Also, when no one is accountable, one can sometimes get more risky recommendations than would otherwise result. This decision must be based upon the nature of the application and the group. In any case, the identification of a member's voting position should only apply to the final evaluation phase of a group process.

3.4 MODERATION AND FACILITATION

In Computer Mediated Communication Systems, aside from message systems, if one wants to conduct group oriented communications, there is still a basic need for moderation and facilitation, just as in face-to-face meetings. However, the nature of leadership in the online environment is different from that in the face to face environment.

In the online environment it is much easier to separate the role of process facilitation from that of content leadership. It is also quite easy to develop a number of different leaders for different areas of a problem.

In the paper and pencil Delphi every contribution first goes to the coordinator of the exercise and then is integrated into a single summary provided to all of the participants. Clearly, in the computer based environment, this is not necessary. Whether or not given contributions need to be screened ahead of time is a function of the application and the nature of a particular contribution. Since the individual members can update themselves on what is new before making a contribution, the amount of duplication is minimized in a computer based Delphi.

For example, it may be desirable to hold certain types of contributions until the group is at a point in the deliberations where they are ready to deal with them. Also, information such as voting results should not be provided until a sufficient number of votes about an item have been accumulated. In situations dealing with very strong controversies, it may be necessary to screen and edit the wording of certain contributions to try to minimize emotional biases and tactics such as name calling and insulting remarks.

While a lot of material in an online Delphi can be delivered directly to the group, the specific decisions on this still need to be made by the person or team in control of the Delphi process. In Computer Mediated Communications, the activity level and actions of a conference moderator can be quite critical to the success of an asynchronous conference and specific guidelines for moderators can be found in the literature (Hiltz, 1984).

There have been many Delphis where the material is summarized based upon the breakdown of the respondents into various specialized expert subgroups or differing interests and perspectives. In the computer based environment it becomes possible to consider multiple group environments. This is where there are separate communication structures or separation of the respondents into sëparate Delphi groups. On top of this

structure would be a higher level one that synthesizes or filters out the reduced set of information necessary to pass between the groups. This does lead to the possibility of very large populations of respondents engaged in common task objectives. A practical example of this is multiple industrial standards groups which must be informed of what is arising from other groups that impacts on their considerations, but do not need to be involved in details of the subgroup deliberations in other areas.

There are many Delphi applications where respondents actually engage in taking on roles (e.g Stakeholder Analysis, Linstone, 1984) to deal with certain situations. This requires moderator supervision and direction. Associated with role playing is the employment of gaming situations where there may be groups in competition with one another and communication is regulated by the "game director" (Hsu, 1989). In the area of policy analysis it could be very productive to allow the subgroups that have agreement about a given resolution to have a private open conference where they can discuss the best possible responses to the material in the main Delphi as a private subgroup. It is also clear that subgroups could be formulated dynamically based upon the content of what is taking place.

Multiple group Delphis in a computer environment is a relatively new potential and there are no hard and fast rules for setting up communication structures in this area. As group oriented Computer Mediated Communication Systems become more widely used, there will be much opportunity to experiment with this relatively new opportunity for structuring communications at both the inter and intra group level.

3.5 STRUCTURE

The heart of a Delphi is the structure that relates all the contributions made by the individuals in the group and which produces a group view or perspective. In a computer based Delphi, the structure is one that reflects continuous operation and contributions. This is somewhat different than the paper and pencil mode where the structure must be divided into three or more discrete rounds. As an example, we will describe potential transformations of two simple structures that have often been utilized in paper Delphis, for use in a computerized environment.

3.5.1 The Policy Delphi

The first example is the Policy Delphi (Turoff, 1970). This is an interesting Delphi structure in that its objective is not to produce a consensus, but to expose the strongest pro and con arguments about differing resolutions of a policy issue. It is a form of policy analysis that provides a decision maker the strongest arguments on each side of the issue. Usually one utilizes as respondents individuals who have the strongest opposing views.

The structure of a Policy Delphi is very simple.

Policy Delphi Structure

TYPE OF ITEM	VOTING SCALES	RELATIONSHIPS
Resolution	Desirability Feasibility	Alternatives
Argument	Importance Validity	Pro or con to a given resolution Opposing to other arguments

In the above structure any respondent in the Delphi is free to add a possible resolution (solution) to the basic policy issue, or to make a pro or con argument about one or more of the listed possible resolutions. He or she can do this at any time. Also, the respondent can vote at any time on the two types of voting scales associated with either of the item types. Individuals may also choose to change their vote on a given item at any time. In this structure the two scales are needed to highlight situations where policy resolutions might be rated in such categories as desirable but infeasible, and arguments may be

rated as important but invalid (others might believe it). When making additions of a qualitative nature, participants must also indicate how that addition is related to the existing items.

The computer's role in the above process is to organize everything so that the individual can follow what is going on and obtain a group view:

Provide each member with new items that they have not yet seen.

Tally the votes and make the vote distribution viewable when sufficient votes are accumulated.

Organize a pro list and a con list of arguments about any resolution.

Allow the individual to view lists of arguments according to the results of the different voting scales (e.g. most valid to least valid arguments).

Allow the individual to compare opposing arguments.

Provide status information on how many respondents have dealt with a given resolution or list of arguments.

The role of the Delphi Coordinator or human facilitator is very minimum in such a well defined structure. The software powers or special privileges that such an individual needs are:

Being able to freeze a given list when it is felt there are sufficient entries to halt contributions, so as to focus energies on evaluation of the items entered to that point in time.

Being able to edit entries to eliminate or minimize duplications of resolutions or arguments.

Being able to call for final voting on a given item or set of items.

Being able to modify linkages between items when appropriate.

Reviewing data on participation so as to encourage participation via private messages.

It is possible to also develop rules to allow the computer to handle some of the above functions. But in terms of today's technology, these functions are still better handled by a human. A group using this structure for the first time should go through a training exercise. The Policy Delphi structure can be designed to be fairly easy to learn and utilize. The use of graphics to support visualization of the structure of the discussion can also be helpful.

The Policy Delphi structure was first implemented in paper and pencil in 1970 and was later implemented in two separate computer versions (Turoff, 1972; Conklin and

Begeman, 1987). It should be noted that the structure of relating items in a Policy Delphi may also be viewed as a representation of a specialized or tailored Hypertext system (Conklin, 1987; Nelson, 1965). Most Delphi designs, when translated to a computer environment, do depend upon semantic relationships among items being established and are utilized for browsing and presenting content oriented groupings of the material. A generalized approach to supporting Delphi relationships within a Hypertext environment may be found in the literature (Rao & Turoff, 1991; Turoff, Rao, and Hiltz, 1991).

Most Delphi structures can be considered to be types of items (i.e. nodes) which have various relationships (i.e. links) to one another. Therefore, it is possible to view a specific Delphi as a particular instance of a Hypertext system. Hypertext is the view of text fragments in a computer as the nodes within a graph or web of relationships making up a body of knowledge. Hypertext functionality is therefore useful for the support of automated Delphi processes.

3.5.2 The Trend Model

This Delphi involves first choosing a specific trend of concern to the group. For example, this might be deaths from AIDS or the amount of life extension expected from a particular treatment. One might include in a single study a set of related trend variables. For the purpose of this explanation we will focus on one trend.

The individual respondents are asked first to make a projection of where they think the time curve will go in the next five years. Then they are asked to list the assumptions they are making and any uncertainties they have. Assumptions are things they think will occur over that time frame and which impact on determining this trend. Uncertainties are things they do not think will occur, but if they did, they would cause changes in estimates of where the trend will go.

Since some peoples' uncertainties are other's assumptions, these are compiled into a list of "possible" assumptions and every individual is asked to vote on each possible assumption according to validity. To accomplish this validity estimation the group may be provided with an anchored interval scale which varies, for example, from "definitely true" to "definitely false," with a mid-point of "maybe." The resulting list of assumptions is automatically reordered by the group validity judgement. The ones the group agrees on as valid or invalid are set aside, and the subsequent discussion focuses

on the assumptions that have an average vote of "maybe". The analysis of the voting has to point out which "maybe" votes result from true uncertainty on the part of the respondents, and which result from wide differences in beliefs between subgroups of respondents.

Clearly in the computer environment, this process of listing, voting, and discussing the assumptions can take place on a continuous basis. The voting serves to quickly eliminate from the discussion those items on which the group agrees. The remaining uncertain items usually are divided into two types: 1) those which can be influenced (e.g. improvements in knowledge about the proper use of condoms), 2) those that cannot be influenced (e.g. hospital facilities in the short term).

In the final stage, after the list has been completed and evaluated, the participants are asked to re-estimate their earlier trend estimate. One could observe that a statistical regression analysis might have produced a similar trend curve. However, the application of such a mathematical technique will not produce the qualitative model that represents the collective judgement of all the experts involved. It is that model which is important to understanding the projection and what actions can be taken to influence changes in the trend or in understanding the variation in the projection of the trend.

There is practically no planning task where the above trend analysis structure is not applicable. In the medical field, for example, considering examining trend curves for the occurrences of certain medical problems and the impact of various treatments is rather broadly applicable. This particular structure has been utilized in a significant number of corporate planning exercises. With graphic capabilities on workstations, it would be quite easy to implement in a computerized version. A similar structure may be applied to qualitative trends made up of a time series of related discrete events. An example would be AIDS cases triggering specific legal rulings and particular ethical dilemmas. The above two examples were chosen because they are fairly simple and straightforward. However, there are literally tens of different Delphi structures that have been demonstrated in the paper and pencil environment (Linstone and Turoff, 1975) and are quite transferable to the computer based environment. Many of these require the ability to utilize graphics to view the complexity of relationships among concepts. Others require extended facilities to utilize generalized Hypertext structures. However, one of the most significant potentials for the automation of Delphi is the incorporation

of real time analysis aids for the interpretation and presentation of the subjective information produced in a Delphi exercise. This will be treated in a following section. It should also be clear from the above examples that there are certain fundamental tools that apply across a wide range of Delphi structures. The ability of a group to contribute to building a specific list, to be able to apply specific voting capabilities, and to be able to sort the list by voting results, represents a set of general tool capabilities. This is the approach we have taken in the development of the EIES 2 system (Turoff, 1991) at NJIT to support a wide variety of applications such as Group Decision Support, Delphi Design, Project Management, and Education (Hiltz, 1986, 1990).

EIES 2 is a general purpose Computer Mediated Communication System that provides many features whereby an individual moderating a specific conference can tailor the group process. The moderator of a given conference can create, at any point in the discussion, an "activity" that may be attached to a comment. These activities accomplish different specialized functions such as list collection and voting. However, the interface to all these activities is the same in the sense that the same basic generic commands apply to any activity. For example, one may "Do" the activity to make changes to it or "View" the results of the activity, regardless of what type of activity it is. The conference moderator has the authority to introduce these activities whenever he or she feels they fit within the current discussion. Also, the moderator may choose to allow or not allow the facilities of anonymity for a given activity or conference.

EIES 2 also provides a general notifications capability that can be tailored to notify the participants in a group process whenever any action occurs of which they need to be made aware. For example, a notification may let the members of a Delphi know when the votes on a specific item are sufficient to allow viewing of the resulting distribution.

EIES 2 is constructed so that any programs or analysis routines developed in any language within the context of the UNIX operating system or a TCP/IP network can be integrated or made available through the EIES 2 interface. The major facility to allow a Computer Mediated Communication System to enhance Delphi processes is to provide alternative structures in the form of a collection of group support tools. The system must also include the privileges for a facilitator or group leader to decide on the dynamic incorporation of these tools in the group process.

3.6 ANALYSIS

A principal contribution to the improvement of the quality of the results in a paper and pencil Delphi study is the analysis that the design and coordination team can perform on the results of each round. This analysis has a number of specific objectives:

Improve the understanding of the participants through analysis of subjective judgements to produce a clear presentation of the range of views and considerations.

Detect hidden disagreements and judgmental biases that should be exposed for further clarification.

Detect missing information or cases of ambiguity in interpretation by different participants.

Allow the examination of very complex situations that can only be summarized by analysis procedures.

Detect patterns of information and of sub-group positions.

Detect critical items that need to be focused upon.

To accomplish the above, there are a host of analysis approaches that come from many different fields. Many of these are amenable to implementation as real time computer based support to a continuous Delphi process conducted via a Computer Mediated Conferencing System. We will briefly address here some of the most significant types of these methods for supporting Delphi applications.

3.6.1 Scaling Methods

Scaling is the science of determining measuring instruments for human judgement. Clearly, one needs to make use of appropriate scaling methods to aid in improving the accuracy of subjective estimation and voting procedures. While most of these methods were originally developed to measure human judgement, they are easily adaptable, in many cases, to providing feedback to a Delphi group on the consequences of the judgements being made by the individuals.

For example, in many cases the appropriate judgement we wish to solicit from an individual is a ranking (i.e. ordinal scale measurement) of individual items. It is comparatively more accurate to ask individuals to rank order items, such as objectives or goals, than to ask for interval or ratio measures. A person can estimate that a particular goal is more important than another one; however, how much more important it is much more difficult to estimate consistently among a group of individuals.

However, a scaling method such as Thurstone's Law of Comparative Judgement

(Torgenson, 1958) can transform individual ranking judgements and produce analytically a group result which is an interval scale rather than a rank ordered scale. Providing the group the results in terms of this interval scale allows the individuals to detect in a much more reliable manner the extent to which certain objectives are clearly distinct from other objectives, and which are considered in closer proximity. Merely providing an averaging of the ranking scale does not contribute this added insight to the group as a whole. Furthermore, standard averaging approaches can lead to inconsistencies in group judgements (i.e. Arrow's Paradox). This can occur when there are disagreements underlying the averaging and when there is a lack of appropriate "anchoring" of the scales.

Standard correlation analysis approaches can be utilized to determine if there are subgroups or patterns of agreement and disagreement that exist across different issues or judgements made in the Delphi exercise. Do the people who feel a certain way about an issue feel the same way about another issue? This type of analysis should, in most cases, be provided first to the facilitator, and that person should decide which relationships need to be passed back to the group. In many Delphis, there are identified sub-groups. A Delphi might comprise people from different disciplines. Do the administrators, researchers, lawyers, insurers, and practitioners have differences in viewpoint that are based upon the perspective they take on a new medical treatment? The utility of these insights needs to be evaluated by the facilitator in the context of the application. With groups that work together over a long term, it might be desirable to provide such an analysis in terms of direct feedback without facilitator intervention.

Scaling methods span a wide range of techniques, from fairly simple and straightforward to fairly sophisticated. An example of a sophisticated approach is Multi-Dimensional Scaling (Carroll and Wish, 1975). MDS allows subjective estimates of similarity between any two objects to be translated into a relative position in a Euclidean space. It provides, in essence, N dimensional interval scaling of similarity estimates. The number of meaningful dimensions found suggests the number of independent dimensional factors underlying the way both the individuals and the group are viewing the similarity among objects. By looking at the alternative two dimensional projections, it is possible to arrive at an understanding of what the dimensional factors are.

The process by which one would use MDS in a Delphi would be to ask for the similarities and provide back the graphical layouts of the alternative dimensions. The respondents would then be asked to try to determine what these dimensions mean or represent. The result is a very powerful technique for potentially exposing the hidden factors a group is using to make judgements about similarities. The question of similarity is one that can be applied to a very wide range of object types, e.g. goals, products, countries, relationships, jobs, criteria, etc. MDS may also be viewed as a form of Cluster Analysis, and many methods in Cluster Analysis (Anderberg, 1973) can also be usefully applied to analyzing the subjective comparison judgements made by Delphi respondents.

When a group is using voting and estimation structures over a long period so that they make judgements about a growing number of similar situations, it is possible to consider the introduction of "scoring" methods (Dalkey, 1977), into the Delphi process. Given later feedback upon the accuracy of estimates or the quality or success of a given judgement, it is possible to consider feedback to estimators on their degrees of "accuracy" or possible biases due to factors such as conservatism. At the point where there are individuals utilizing Delphi techniques on a continuous basis, it will be possible to conduct the sorts of investigations needed to develop this particular area as a decision aid.

Designing a Delphi, whether via paper and pencil or on the computer, does include the process of designing a survey. As such, all the guidelines on good survey design and all the analysis methods that have been developed for analyzing survey data are potentially applicable to a Delphi. There is, however, a fundamental difference in objectives, which determines how one employs a given method, and whether it is applicable in a given situation.

Most scaling methods were evolved to aid in making an assessment of a human judgement with the premise that one is measuring a stable and constant quantity. One's intelligence or personality would not be affected or changed as a part of the measurement process. The goal is to discover biases and inconsistencies and to produce more accurate measurements. In the Delphi process, however, we are interested in informing the respondents about what they are really saying, and how it compares to the group as a whole. We are also interested in promoting changes in viewpoints and the other items we measure, if it will promote reaching a superior group view of the

situation. We are also interested in detecting and exposing hidden factors or relationships of which the group may not be completely aware. With this in mind, one has to take special care that the use of these analysis methods does not convey a false impression of finalization in a group view.

Related to scaling is the area of Social Choice Theory, which provides alternative methods for the summarization of voting processes (Hogarth, 1977). The use of multiple methods of viewing the summarization of a given voting process can be useful in preventing a group from placing an over emphasis on a single voting result.

Probably the most important single consideration in the past that has prevented the incorporation of many of the approaches discussed here is the difficulty of educating the respondent in the interpretation of the method when the respondent is involved in only one short term Delphi process. With the potential that Computer Mediated Communications offers for long term continuous use by groups, it is now possible to consider incremental training for individuals to gain an understanding of the more sophisticated methods.

With the appropriate use of scaling methods it becomes possible to establish that individuals will mean the same thing when they use terms like: desirable, very desirable, likely, unlikely, agree, strongly agree, etc. It becomes possible to determine which alternatives are truly similar and which are distinctly different. Scaling methods, in essence, serve the objective of eliminating ambiguity in the judgmental and estimation process of a group.

3.6.2 Structural Modeling

The term Structural Modeling (Lendaris, 1980; Geoffrion, 1987) has come to represent a host of specific methods that have the objective of allowing an individual to express a large set of independent relationships and judgements which the given method utilizes to produce a "whole" model of the "system" being described. In computer terms, these are methods that allow a user to build a model of a situation without having to program or go through the use of experts in modeling and simulation. These methods vary from ones that provide a simple static relationship model (e.g. Interpretive Structural Modeling, Warfield, 1974), to more dynamic probabilistic and time varying models

(e.g. Cross Impact, Time Series Regression, etc.). Just about any technique that organizes data into some sort of framework is a candidate for falling under the rubric of Structural Modeling. This includes Decision Trees and Payoff Matrices.
The objective of these approaches is to allow participants as individuals or as part of a group, to contribute pieces of a complex situation and to be provided a composite model. For example, in Interpretive Structural Modeling the individual is asked only to make a series of judgements about each two components of a model (such as two goals) with respect to whether they are related. The resulting complex network of relations is analyzed to collapse the network to a hierarchy of levels utilizing the existence of cycles within the network to make that simplification. The result for the individual or group is a set of levels or clusters of the objects which infer a relationship of higher to lower levels. This provides a graphical representation of the binary judgements made about each set of objects, taken two at a time.
The Cross Impact type model allows individuals to express probabilities of occurrence for a series of events, and conditional probabilities based upon assumptions as to which events will or will not occur. This is used to construct a quasi-causal model that allows participants to then vary the original estimates of individual events and see the consequences on the whole event set.
An excellent example of structural modeling to determine the important relationships and impacts on changes in medical care policies may be found in a recent article by Vennix et. al. (1990). This particular example is based upon the specification of negative and positive feedback loops. The development of the model was arrived at through the joint use of a paper and pencil Delphi and follow up face-to-face meetings.
All these techniques may be used in a Delphi process to help a group to develop a collaborative model of a complex situation. This is one area where the merger of the Delphi process and the computer presents a unique opportunity for dealing with situations of unusual complexity. More often than not, the individual experts who can contribute to building a complex model are geographically dispersed, and the effort to derive and improve such models is one that needs to take place over an extended period of time. In other words, improvement of the model has to be based upon feedback from its performance and incremental refinement.
A recent experiment (Hopkins, 1987) produced the very significant finding that it was possible to distinguish the degree of expertise an individual had about a complex

situation by the measured richness of the models that were specified by each individual. This finding suggests the possibility of incorporating automated procedures for rating potential quality or inferred confidence in the contributions made by various individuals. This possibility deserves further investigation, as it would obviously provide improved models with a reduced communication load requirement.
Developing all the structural relationships in models of symptoms, tests, diagnosis, and treatment is an obvious area for the application of structural modeling. Appropriate techniques can be utilized on the computer to allow individuals to visualize the structures resulting from their contributed relationships and examine that structure for consistency. At the group level the same methods can be used to examine composite models for consistency and feed back inconsistencies for further refinement. Individuals are good at estimating individual relationships, but they are not always able to maintain consistency in developing complex models. The problem is compounded for group efforts.
A group can improve the nature of a model only by first seeing the results and consequences of the current design. Model building is a long term incremental process. The proper integration of Delphi methods, Computer Mediated Communications and Structural Modeling methods makes possible effective large scale modeling efforts not otherwise currently doable.

3.7 DELPHI, EXPERT SYSTEMS, GDSS, AND COLLABORATIVE SYSTEMS

The concept of an Expert System is to somehow capture the knowledge of a group of experts and store it in a computer for utilization by non-experts. The incorporation of the Delphi method in computer environments makes possible a number of significant refinements of this objective and some fundamental possible changes to the nature of Expert Systems.
The common approach to the development of an expert system is to achieve agreement among all involved experts before the actual coding of the knowledge base is performed. At present this is accomplished by a knowledge engineer or team of knowledge engineers, who must interface to a team of domain experts. Besides being time consuming, the fundamental flaw in this approach is that even within scientific and/or engineering fields, there is incomplete agreement among experts. Furthermore,

agreement and disagreement are evolving properties that change dynamically over time. The Delphi method may be viewed as an alternative approach to collecting and synthesizing expert knowledge. In fact, within the current terminology, the design of a Delphi is in fact the design of a knowledge base or structure for putting the collected information together. It has also been an important objective of Delphi design to capture disagreements as well as agreements.

Another potential problem area is that experts concerned with a common problem can be in conflict. For example, design, production, and marketing professionals can have severe conflicts about the properties of a potential new product. Different medical researchers have different views about the most promising directions for research. Some of the problems addressed here have been investigated in the work on "Multi Expert Knowledge Systems (MKS) (LeClair, 1985, 1989). LeClair's work represents one of the few in depth approaches to incorporating the knowledge of disagreeing experts into the same system. However, this work still assumes the final system no longer incorporates the humans, but only their knowledge.

On the other hand, the view that we believe is the most promising is an objective for "Collaborative Expert Systems," where the experts are provided a knowledge structure (a Delphi Design) that allows them to dynamically contribute their knowledge to the system and to modify and evolve the system, over time. Clearly, such a system is one which the experts must desire to use for themselves as well as a tool for others who need their knowledge. This is the situation where the experts are both the creators and the users of the resulting expert system.

Without the above form of expert systems, the only feasible systems are those that restrict themselves to well established rules and agreements. In our view, the future of expert systems lies in their ultimate ability to be utilized by working groups of experts as a tool for collecting and assessing their collective knowledge about their work.

The current approach to expert systems through the use of knowledge engineers has been recognized as the chief bottleneck to the creation of these systems (Welbank, 1983; Waterman, 1986) for four main reasons:

Human expertise is usually complex, undocumented and consists of many different types and levels of knowledge (e.g. casual knowledge, common sense, meta knowledge, etc.)

Different experts may solve the problem differently and therefore may argue or even criticize one another on the method used.

There often exists a communication barrier between the knowledge engineer and the experts. The knowledge engineer is not an expert on the area and many experts do not understand their own problem solving process. As a result, many details and complications of the reasoning process may be ignored or obscured.

Motivation for the expert is often lacking because the results are often delayed or are not intended to benefit the expert.

Many of these problems can be overcome if one can develop collaborative design systems that focus on allowing a group of experts to develop their own expert system in an evolutionary manner and as a group oriented aid to their own work. The evolving system could also be tapped by non-experts for use. In that mode it would be considered by the experts as an aid to disseminating needed information to a wider circle of users and freeing the time of the experts for more difficult problems.

A collaborative expert system has to deal with at least four types of knowledge:

Deductive reasoning as represented by rule based models.

Inductive and intuitive reasoning representing experience on the part of experts.

Objectives, Goals, and Vested Interests which are viewpoints of experts in given circumstances.

Values and Beliefs which often underlie judgements about viewpoints.

The first two types have been typical of current expert systems. The other two areas have largely been the domain of Group Decision Support Systems, Delphi and Nominal Group Techniques, decision and utility theory, and psychological measurement methods. All four of these types of knowledge in a collaborative expert system must handle disagreement among the participants.

3.7.1 The Deductive Level of Disagreements

At the predicate logic level, experts may disagree about both the predicates to use and the rules that are valid in the real world. A well designed knowledge acquisition and expert environment should permit experts to "speak their mind" and not limit them to a preconceived vocabulary. It is therefore necessary that the accumulation of the vocabulary for specification of the rules be an integral part of the collaborative process. Even if experts agree on a basic vocabulary, they often disagree about subtle details of a representation. This problem occurs whenever there are several possible reference frames, a situation which is well documented in the literature (e.g. Sondheimer, 1976). Unfortunately, at the current state of the art, two relations with different numbers of arguments are treated by logic programming environments as being two completely different entities.

One approach to this problem is to allow each member of a collaborative group to construct and tailor their own knowledge base and then to superimpose an analysis system for determining various types of agreement and disagreement. There are various weighted voting procedures (Shapley and Grofman, 1985) and scaling methods (Torgerson, 1958) that are promising for analysis of this situation. Weights have been used in some expert systems (e.g. Reboh, 1983). When such information is being accumulated over time then there are various "scoring" approaches (Dalkey, 1977) that may also be employed and coupled with "explanation based learning" approaches (Pazzani, 1988). Early work with the Delphi method indicated that even experts in a given area differ in expertise in various sub-domains and that the greatest improvement in accuracy of estimates was obtained by weighing estimates by this type of difference (Dalkey, 1970).

3.7.2 The Inductive Level of Disagreements

One of the major problems in designing knowledge representations that reflect common sense models of the world is that the world is not a discrete and well specified place. In fact, the world is quite vague and ambiguous. Ambiguity is the key property that most people have to deal with in reaching conclusions and decisions (Daft and Lengel, 1986). Ambiguity results from differences in concepts (e.g. "expensive") among different people and, in this context, from the collaborative process itself. In many cases the problem of ambiguity can be structured as the degree to which an object "more or less belongs" to a class. Fuzzy sets (Zadeh, 1965; Klir, 1988) are a generalization of standard sets that allows for degrees of membership. One approach to this problem is to utilize fuzzy set theory to represent the types of ambiguity that result from intuitive thinking.

The major research issue in this area is to develop methods for accurately combining multiple judgements and resolving disagreements about estimates of degrees of membership in fuzzy set relations (Stephanou and Sage, 1987). In this instance, scaling methods seem particularly appropriate. Humans are good at ranking (object A belongs more than object B) but not good at direct estimates of correlation factors needed for fuzzy set relationships. However, various scaling methods can be used to convert a collaborative set of ranking measures to interval or ratio scales.

Another approach is to incorporate multivalent and fuzzy logic (Dubois and Prade, 1980) into any model framework where the expert group is building the relationships. An example of degrees of truth and the resulting treatment of logical inference from a fuzzy perspective may be found in Baldwin (1981).

In essence, the problem is the recognition that models that capture intuition have to capture the structure of disagreements. A result is no longer true or false, but possibly a little of both. Rather than a group process being dedicated to eliminating disagreement, the objective is to capture it, quantify it, and integrate it into the collective model. There has always been a bias that disagreement has no place in the result of a scientific process. Because of this we can become blind to the forcing of unwarranted consensus. It would be a far more realistic view of the world to recognize the necessity for disagreements and "fuzzy" relationships as a fundamental part of any model meant to reflect the collective intuitions of a group of experts.

3.7.3 Goal and Value Disagreements

This is the area that is typically included in applications of Group Decision Support Systems. While there are certain specific approaches (e.g. Stakeholder analysis) for eliciting this type of information, the current state of the art is largely the use of human facilitators to guide the group process for the treatment of this type of knowledge. The fundamental issue of how far one can go in the process of substituting computer facilitation for human facilitation is very much an open issue. Earlier experiments in this area (Turoff & Hiltz, 1982; Hiltz et. al., 1986, 1987) showed that under some circumstances computer facilitation can degrade the performance of the group.

The approach that seems to be the most promising is to evolve a collaborative expert system that would be used to guide the meta group process. This would suggest to the group at what points in the activity they should shift the nature of what they are doing. However, such a facility would also be tailorable by the group so that it can gradually adapt to the preferred group process. Such a system would have to employ "default reasoning" approaches (Post, 1990).

As can be seen, there is no fixed dividing line between such areas as Delphi, Computer Mediated Communications, Group Decision Support Systems, and now Expert Systems. The concept of "collaborative expert systems" is really based upon the foundations established in each of these other areas. Subjective estimation, collaborative judgement formulation, and voting are strongly related support areas that also contribute to the potential for design in this area.

3.8 CONCLUSION

Delphi, as a tool, has reached a stage of maturity in that it is used fairly extensively in organizational settings in either the paper and pencil mode or in combination with face-to-face meetings and Nominal Group Techniques. Since most of these exercises are proprietary in nature there is not much of this activity reported in the open literature. The one exception to this is the applications in the medical field which are in fact actively reported and documented (Fink, Kosecoff, Chassin, and Brook, 1984). This clearly is a result of the growing need to formulate collaborative judgements about complex issues that are associated with the production of guidelines on medical practice and decisions.

Computer Mediated Communications has also seen some very significant applications in the medical field with respect to the formulation of collaborative judgements. One of the most significant to be reported in the literature was the use of leading researchers in Viral Hepatitis to review the research literature and update guidelines for practitioners (Siegel, 1980). While this was not run in an anonymous mode, it had all the other aspects of structure necessary for a dozen experts to deal with some five thousand documents and reach complete consensus on the resulting guidelines.

Another CMC application that had Delphi like structuring with Anonymity was a Group Therapy process to aid individuals in the cessation of smoking (Schneider, 1986;

Schneider and Tooley, 1986). A general review of CMC applications in the medical field can be found in Lerch (1988).
However, there is yet to be a true merger of Delphi with Computer Mediated Communications. It is only now that the technology is becoming generally available to support the high degree of tailoring necessary to dynamically structure communications within a single conferencing system (Turoff, 1991). Most conference systems, to date, have only represented single design structures with very little control available to facilitators and moderators of discussions. Also, the general lack of graphics has placed a considerable limitation on just what Delphi techniques could be adapted to the computer environment. The merger of Delphi and Computer Mediated Communications potentially offers far more than the sum of the two methods.
Long before the concept of Expert Systems it was known that statistical factor models (Dalkey, 1977) applied to a large sample of expert judgements could produce performance that was consistently in the upper quarter of the performance distribution curve. Such models did not suffer from "regression to the mean" and could result in matching the best decisions by the best experts in the group. Expert Systems is really the emergence of tools to allow this to be done on a fairly wide scale. However, the results of Expert System approaches, as currently practiced, are never going to do better than the best experts.
The merger of the Delphi Method, Computer Mediated Communications and the tools that we have discussed opens the possibility for performance of human groups that exceeds the composite performance curve. We have termed this phenomenon "collective intelligence" (Hiltz and Turoff, 1978). This is the ability of a group to produce a result that is of better quality than any single individual in the group could achieve acting alone. This rarely occurs in face-to-face groups.
A recent experiment in utilizing human judgement in conjunction with the types of models that are used in Expert Systems confirms that this is in fact possible (Blattberg and Hoch, 1990). There has been too much attention in recent years to utilizing computer technology to replace humans and far too little effort devoted to the potential for directly improving the performance of human groups. This can be achieved through integration of computer based methods and the concept of structured communications at the heart of the Delphi Method.

CHAPTER 4

ARCHIVE SYSTEM

An introduction to principles of organization and description used in archives
A first step towards acquiring the knowledge and skills needed to provide access to the rich cultural heritage information in archival collections.
Links to additional resources for further archival training such as workshops, readings, professional organizations, archival education programs and conferences.

4.1 Basic concepts and principles of archives and records management

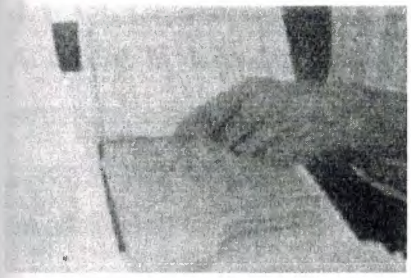
Through the centuries, the following THREE FACTORS have shaped the concepts, principles and techniques which records/archives managers use to carry out their responsibilities:

4.1.1 The Inter-relatedness of records

Because records are the documentary by-products of work or life processes, they are, like individual frames of motion picture film, organic bodies of related material which cannot be used in isolation or separation from one another without loss of integrity and meaning. They are unselfconscious in that they are naturally occurring contemporaneous and candid documents, as opposed to individual documents created intentionally for the purpose of 'history'.

4.1.2 The central importance of context

Records draw their significance from their context. That is, they are valued or useful only in groups and only in relation to the activities and purposes for which they were created and used. Thus records/archives managers must accurately identify and explain both the context of origin/creation and the context of use/custody and maintain the records in a way that preserves their original character and relationships as 'bounded entities comprising content, structure and context'.



Records rescued from past neglect often come to the archives in a disordered mess. Archivists must spend time carefully examining material to identify and restore its original provenance and order before it can be used by researchers.

4.1.3 The function of records as evidence

As we have seen, records represent or stand for human experiences, transactions, activities or accomplishments. The records designated as the 'official' or 'record copies' of documents have been selected to endure as unique testaments, all other duplicates, whether exact copies and different formats having been destroyed. They provide objective 'proof' that something has happened or been agreed to by consenting parties and as such have an integrity that must be protected and preserved by responsible and continuous custody and properly authenticated if that chain of responsible management is questioned.

Essential characteristics of recordness

What distinguishes records from other information entities is summed up in the term 'recordness'- an elusive quality usually represented by the six characteristics described below.

4.1.3.1 Recordness requires that records are:

4.1.3.1.1 Complete

A record is considered complete when it is a finished, bounded entity comprising structure, content and context and when it has the following elements: date(time and place of creation, transmission and/or receipt); originating address, an author/compiler, an addressee/recipient, title or subject accompanying its content/message.

4.1.3.1.2 'Fixed'

Records are information presented in a static form. The act of recording 'freezes' the relationships among the information elements and embeds them in a structure, which can be replicated or re-constituted. For example: data elements within a dynamic or 'live' database are not records until a transaction incorporates such data elements (content) into a meaningful and re-creatable format (structure) along with essential specific contextual information (context). The resulting record is a 'bounded entity' - a metadata encapsulated object (MEO) which may be retrieved and re-presented for human use in electronic form (screen display) or hard-copy printout (paper or film). Although a record is a 'fixed' object, its role is dynamic, not static. Subsequent records and business activities change its relationships, significance and meaning as time passes.

4.1.3.1.3 Organic

Records are the natural output of work processed; therefore records are only meaningful as a sequence of transactions. Each record is related to or is a consequence of some preceding document, with the matters documented by the former further explained or dealt with in the latter.

4.1.3.1.4 Contextual

Records derive their meaning, and therefore their usefulness or value, from their context. Because they are created, organised and used in the conduct of normal business by a particular entity, they reflect the purposes and the activities of that entity over time.

Thus, to use/appreciate the records one must understand their function, control system, relationships and pattern of use in the workplace; correspondingly the records comprise the evidence of the policies, activities and transactions of that creating entity.

For this reason, records are managed not as individual items, but in aggregates known as series, which are organic 'flows' of material created and structured by the normal course of work activity.

4.1.3.1.5 Authoritative/'Official'

Records created as documentation to support ongoing work or business activity have a status as 'official' evidence of those decisions and actions undertaken 'in the normal course of business', whether that 'business' is commerce or art or literature or just living. That is, they embody the full and unchallengeable authority of the author - be it an individual or an organisation and its officers.

4.1.3.1.6 Unique

Unlike books, journals and other published material, records appropriately maintained in context are always one-of-a-kind. That is, any record or group of records with its sequences and interrelationships is unique, despite the fact that there may be duplicate copies of some or even all individual records held elsewhere. The point is that each copy will have a different relationship to the conduct of business and to those who authorised or participated in it. However, in cases where duplication is extensive, it is usual for recordkeepers to minimise the number of duplicate series of records in their care by designating the most authoritative set as the 'official record' for retention and directing those of lesser importance to be destroyed.

Foundation principles for organising and keeping archives

With these three underlying factors and related characteristics providing the intellectual background, let us now explore the Foundation Principles of modern archives management.

4.1.3.1.6.1. The principle of provenance

This fundamental principle of grouping requires that records be organised and maintained according to their transactional origin or source. In short, records originating from one office or individual form a distinct body of material, which is to be

Records must be managed and accessible within an archive so that their provenance, original order, and chain of responsible custody is established and maintained
kept separate and inviolate. It must not be intermingled with records of other 'parentage'. Records from the same origin came to be known as 'fonds' and the principle as Respect des fonds, reflecting its French popularisation and as provenienzprinzip in German. The thinking behind this principle reasons that for records to serve as evidence, they must be traceable to their source and be shown to reflect their contexts of origin/creation and initial or primary use.

Practically speaking, this principle was adopted to preserve the chain of accountability within the ever-growing number of records documenting similar functions and activities produced by growing bureaucracies. It was important to know who was initially responsible in each transaction and to maintain an authoritative custodial lineage

4.1.3.1.6.2. The principle of original order

The Prussian (later German) archivists of the mid-19th century expanded the influence of the office of origin by developing and establishing the related Principle of Original Order or Registratorprinzip in German. This maxim stipulates that records are to be maintained in records/archives repositories in the same scheme of order and with the same designations they received in the course of the business of their office of origin and primary use. Again, the emphasis was on establishing the authenticity and integrity of the record as evidence of work processes and activities in context.

4.1.3.1.6.3. The chain of responsible custody

Completing the trilogy of context and process oriented principles for records/archives management, we come to the Principle of Continuous Custody. Articulated and popularised by the English archivist, Sir Hilary Jenkinson, this principle focuses upon the role of records/archives as evidence and maintains that evidential integrity can only be ensured when we can trace 'an unblemished line of responsible custodians'. By responsible, Jenkinson means committed to the 'physical and moral defence' of the archives. These phrases embody the responsible manager's obligation to ensure both the physical security and the intellectual integrity of the records as evidence. This faultless lineage is, in Jenkinson's view, a reasonable guarantee that the records have been kept without damage, alienation, improper or unauthorised alteration or destruction.

Protection of the essential role of records as evidence is the wellspring from which all concepts, theories, principles, policies and practices of sound recordkeeping arise. Any new approach or technique for managing records must adhere to them or acknowledge their centrality and fully justify any modification or proposed departure from them.

4.1.3.1.6.3.1 What goes on in the archives?

Because of the power and importance of records as a resource, managing records over time is a necessity. Management means imposing a regime which influences the control, accessibility, disposal, and storage of this irreplacable evidence and which manages itself effectively.

Regimes adhering to these core functions ensure that reliable records of the highest quality and integrity are available in a timely fashion for authorised use at the right price. These processes further guarantee that the best of the records continue to be available effectively and efficiently as part of our cultural knowledge base.

4.1.3.1.6.3.2 CADSS

The functions embodied in the acronym CADSS (for Control, Accessibility, Disposal, Storage and Sustain) make up the core of all information management activities.

4.1.3.1.6.3.3 CONTROL

The CADSS management model begins with C symbolising the function of Control. Whilst the word 'Control' also conveys the overall goal of records/archives management, Control as a discrete function includes all recordkeeping activities required to:

bring a record into existence as complete, integrated entity that can serve as reliable evidence of the acts to which it attests;

identify the physical and intellectual attributes of a record's content, structure and context;

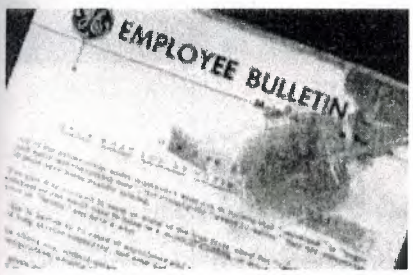
devise/identify and assign a physical 'address' for the record so that the record can be safely stored and efficiently retrieved;

articulate/represent and document the attributes and 'address' above as an integral part of the record and of the information systems controlling access and retrieval of the record and/or of information about the record

Machine-readable records such as video and audio tapes are more complex to describe than paper-based materials because understanding and accessing them needs special machinary. Documenting how these records were made and what technologies were used in their making is an integral part of the 'Control' function of record keepers.

In this photo, an archivist makes notes about the content of an oral history tape, relating it to numbers as registered by a counter on the tape recorder. Future listeners should be able to find particular content without listening to the whole oral history.

In creating/custodial offices, Control would include all tasks associated with creating and/or receiving and organising records physically and intellectually for initial use. If there is a centralised or coordinating entity such as central registry, records management and/or information services with this responsibility, there is a chance for cohesion and continuity.
However, the likelihood that these entities have been discarded or ignored in the technologically driven decentralisation of management and concurrent elimination of clerical support (more chiefs, less indians) means that there may be no records system-wide coordination over such critical factors as file content, structure, titling or indexing/access points, nor would any given record necessarily have an inviolable, integral link with its context of creation.
You never know what you might encounter in a body of records. Ham sandwiches, cockroaches, peanuts, well chewed gum, even mummified birds have been uncovered when arranging and describing abandoned masses of files.
However in this case, the lollypop is not detritus, but an integral part of the record. This special circular was sent to employees by a trade union seeking members support against the company.



In traditional archival repositories, Control embraces a range of activities known as accessioning and arrangement and description. These processes document the nature and origins of the material and explain them to prospective researchers.
In many cases archivists must conduct extensive research to recover or re-construct information needed to explain records that was lost long ago. In this work, they may employ investigative principles and techniques similar to those used by archaeologists.

4.1.3.1.6.3.4 ACCESSIBILITY

The function of Accessibility covers all activities associated with determining, administering and facilitating access to and use of records/archives. Accessibility may involve the acquisition or design and operation of specialist facilities, services, expertise and information sources to ensure that:

laws, regulations, conditions and terms of access to records/archives are suitable, authoritative, documented, disseminated and properly administered;

information about the records/archives, about obtaining access to them and about using them effectively is accurate, understandable, timely and readily available to authorised users;

prospective users and their uses are appropriate, authorised and documented;

the records/archives are retrieved, used and returned to safe custody in a timely manner.

Providing access to images can be difficult as pictures do not translate well into words and producing full size photographs of images is costly. One solution which achieves access and preservation objectives is to copy the most important images, creating a security negative for preservation and a miniature contact print to include with the finding aid in bound form or in a loose card file as shown here.

With the advent of computers and scanning technologies, many of these manual files can be scanned and made accessible electronically. Image databases can provide digital access to important photo collections either on site or over the Internet

In creating/custodial offices, Accessibility is represented by the systems and practices that govern which staff may see and use records and the mechanisms for controlling and documenting authorised access and use. Common activities include checking passwords and clearances,monitoring record movements documentation and auditing use.

In traditional archives, the activities of Accessibility are called 'reference services' and centre around a designated reading or research room facility.
Accessibility may also involve the creation of more detailed or subject oriented finding aids such as indexes or special guides or lists. These tools create additional points of access (by subject, name of participant, date, type of record, geographic location) which complement the basic structural finding aids based upon provenance.
Client-oriented training and education to promote the use of archives or to develop research skills is also included in the accessibility function

Providing a suitable environment for research work is a continuing challenge. This supervised reference facility is spacious, quiet and well furnished for comfort and efficiency.
In the public sector, both archives repositories and creating/custodial offices also have a general responsibility to make information and/or records available to the public under such legislation as Freedom of Information, Privacy and Archives or Public Records Acts.
Popular finding aids, such as this one , are paying a high price from overuse. Part of reference management is to prevent such occurences.

4.1.3.1.7 DISPOSAL

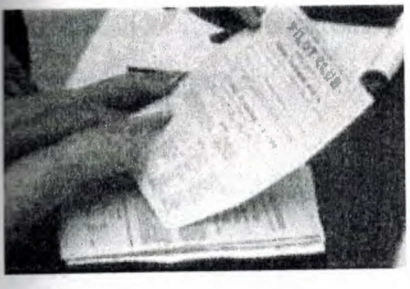
The Disposal function incorporates all activities involved in:

identifying and documenting/surveying organised activities and records making/keeping systems within a designated universe of records keeping responsibility;

determining what documentation reflecting organised activity within a designated record making/keeping system should be retained and, conversely, what should be destroyed;

authorising when, where, how and by whom these decisions will be implemented and documented;

providing advice, mechanisms, facilities and documentation for systematic, secure and accountable disposal processes and outcomes.



Properly designed disposal instructions inform employees of how and when to close files, of where and how long to store them, and ultimately, whether they are to be destroyed or preserved indefinitely as archives.

In creating/custodial offices, Disposal embraces activities such as:

conducting records use/location/activity audits;

providing information about records use/location/activity for disposal decision-making;

drafting or responding to proposed disposal decisions and recommendations;

approving, implementing and reporting disposal actions.

In traditional archives, the Disposal function is generally manifest in activities to identify and select or appraise/evaluate records of enduring value, but may also include the disposal and de-accessioning of unwanted material following appraisal or subsequent re-appraisal.

4.1.3.1.8 STORAGE

The Storage function is largely concerned with the physical preservation and care of records and archives through the use of archivally sound and/or appropriate :

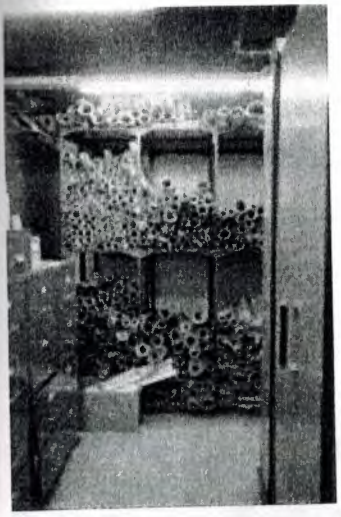
recording technology and media, packaging components/supplies, storage equipment, facilities, macro- and micro-environments;

handling procedures during retrieval/refile, use, copying, display, transfer and transport;

macro-preservation actions including:

- risk assessment and minimisation;
- preventative and protective intervention activities;
- disaster response and recovery planning;
- collection and environmental stability monitoring; and
- informational copying/media migration;

micro-conservation treatments to stabilise, repair, strengthen and/or protect individual documents or series



Training programs to educate record creators and support staff in the proper care of long-term records can help eliminate destructive practices such as those evident in this nightmare storage room.

What threats to record preservation and integrity can you see in this picture?

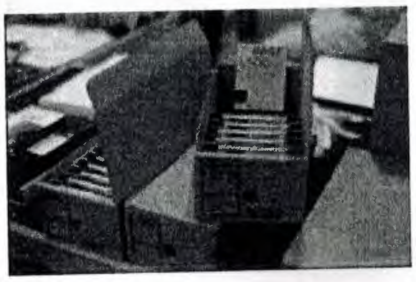
4.1.3.1.9 Check Answer

Once cleaned and de-acidified, single, oversized items such as plans, posters, prints, and drawings are ideally stored inside clear polyester 'envelopes', easily constructed from conservator-approved materials as shown in this photo. Known as encapsulation, this procedure protects items from dust and handling

Within creating/custodial offices, Storage can be badly fragmented. Decisions affecting the choice of record making technology, media, components/supplies and equipment are frequently divided among the Information Systems or IT services section, purchasing, central records/registry (if it exists) and individual office managers of decentralised records systems.

Standards and procedures for records and file maintenance and handling can be chaotic if no central policy or coordinating responsibility exists.

In traditional archival repositories, the Storage function may be shared between two organisational entities- program administration which frequently manages the plant/facilities and technical and/or preservation services, which may also include centralised microfilming, photographic and electrostatic copying.



Specialist forms of records require appropriate packaging to facilitate access and ensure preservation. Here, audio tapes are boxed in acid free cartons to protect them from dust

4.1.3.1.10 SUSTAIN

The final CADSS letter represents Sustain, which is used here as a synonym for management - the function that sustains the recordkeeping regime as a viable and effective component of its host organisation.

Whether a particular operation is large or small, all professional recordkeepers must fullfil their management responsibilities to acquire and deploy valuable resources and to get work done productively, effectively, harmoniously. Know what ensuring essential evidence through effective recordkeeping requires and how it can be achieved in various contexts.

Set realistic written objectives which complement and support the overall purpose and strategies of your host organisation.

Identify others who require essential evidence in their work and involve them meaningfully and appropriately in the process of setting and achieving these objectives.

Acquire and deploy resources (staff, facilities, funds) to meet these objectives.

Use the agreed upon objectives and distributed work tasks as the focus to encourage cooperation/collaboration and achieve productivity (management by objectives).

Assess the quality and quantity of progress and end results (evaluate performance).

Obtain and disseminate essential knowledge, skills, techniques, attitudes on a continuing basis.

Recordkeeping staff are essential to the smooth running of an organisation and require appropriate training and resources to effectively fulfill their roles.

Because recordkeeping regimes must establish and operate effectively across the whole organisation, it is essential that recordkeeping professionals understand and learn to enhance the influence and, thereby, the effectiveness of their recordkeeping regimes. The pathway to success is smoother when one or more of the following conditions exists or can be created:

A workplace culture that fosters productive, professional relationships and welcomes diversity.

A constituency (or at least those members which are themselves powerful or close to power) which knows/ understands/ appreciates the function of recordkeeping.

A host body which itself employs the regime for a full ranges of recordkeeping services, including protection of its own records with enduring value or archives.

An integrated archives/records management program which provides services to meet the host business, regulatory and cultural/historical recordkeeping needs.

An administrative placement and structures that facilitate ready access to key decisionmakers; are linked to powerful units with authority across organisation and/or are not too far down the chain of hierarchy.

A host body with a cohesive focus on recordkeeping, not just on administering a 'heritage' or 'culture'. For example, archival collections may be administered as cultural heritage, along with a museum, art gallery and/or library materials. When archives are administered strictly as cultural objects split off from the recordkeeping regimes that generated them, those powerful organic relationships that link them to ongoing management effectiveness and regulatory accountability are broken. In such cases, archives may appear restrictive and mundane in comparison with more accessible, visual, understandable and, therefore, sexier, more exploitable art and museum material.

Visibility and networking are crucial to recordkeeping effectiveness. As the sign attests, the archivist of this college has positioned the archive for optimum contact with the powerful.

4.1.3.1.11 Recordkeeping regimes

The professionally managed recordkeeping regime = A rewarding investment

Global diversity and complexity is placing greater and greater emphasis upon recordkeeping systems. Good recordkeeping doesn't happen automatically. The design and operation of recordkeeping regimes ie. the programs for making and managing records, requires specialist professional knowledge and skill. And, as with all worthwhile enterprises, you must invest appropriate resources to achieve effective results.

4.1.3.1.12 Where we work: The recordkeeping context

As stated earlier, everyone needs and keeps records, though some do it more extensively and formally than others. Generally, recordkeeping specialists do their work within public or private sector organisations as part of work units bearing some variation of the title: Archives and Records Management Services.

4.1.3.1.12.1 Public sector placements

A public recordkeeping authority that oversees the capture and maintenance of evidence on behalf of 'the people' performs a duty of care that requires objectivity. Thus it may be established as an independent body, or, provided regulations protecting its integrity are present, it may be part of a larger agency administering centralised management or heritage responsibilities.

Independent Body - Statutory authority or corporate entity, often with an advisory board composed of public 'watchdogs', industry experts and stakeholders.

Part of larger agency - recordkeeping regime reports to:

Multi functional cultural or heritage department

Administrative services department

Library or historical society or museum

Department of State or 'secretariat' equivalent

Office of Chief Executive

4.1.3.1.12.2 Private sector placements

Generally recordkeeping regimes in private enterprises are established to serve the mission of that host body and serve business and regulatory requirements as their first priority. However, most private enterprises now recognise corporate good citizenship as a vital and fragile business asset. As a result, many are relying more on their recordkeeping regimes, particularly their archives, to provide long-term evidence of their societal contributions. Mirroring their organisation's primary evidential concerns, most recordkeeping regimes in the private sector are 'headquartered' in one of the following areas:

- Multi functional administrative services
- Legal department
- Secretariat
- Public relations/Advertising
- Research and development
- Corporate information services, including information systems and information technology

Whether public or private, effective recordkeeping regimes must always be centrally designed and coordinated, but may be decentralised in their implementation and daily operations.

4.1.3.1.13 Type and functional emphasis

The tangible features of the program itself reflect the enterprise that hosts the recordkeeping program. Looking at the categories of human activity below, it is clear that there would be considerable differences in each sector's requirements and use of records. For example, private sector mining is less records intensive than medical services or social welfare.

Categories of human enterprise

Social quality sector enterprises - may be public or private or both.

The recordkeeping program of a fast food company will differ from that of a church; a school will have different records and emphases that a bank and so on.

Some programs serve a single organisation; others collect materials from many different sources. Those who manage the records of a host organisation are referred to as in-house or institutional recordkeeping regimes. Those that receive the inactive records/archives of a number of different bodies are known as collecting archives. Some programs may combine elements of both collecting and in-house work and are characterised as comprehensive recordkeeping regimes.

Increasingly in-house and comprehensive programs are utilising the Records Continuum regime management model to manage records from conception to untimate disposition; whereas collecting programs are more historically orientated and offer repository services exclusively for archival materials. Regardless of whether the type of regime is institutional/in-house, collecting or comprehensive, each one will emphasis different functional aspects of recordkeeping. For example, an institutional regime may be more focussed upon recordkeeping to achieve ongoing business objectives; another might be intent upon addressing legislative and regulatory requirements; a third, usually a collecting regime, might be designed more or less exclusively to recover and preserve evidence of the past. However, it is not unusual for a comprehensive program to be involved in all three, in varying degrees.

Researchers from Monash University's Records Continuum Research Group have developed a checklist of features that characterise a fully competent and effective recordkeeping program.

So far the Group has concluded that the components of accountable recordkeeping include:

4.1.3.1.14 Accountable recordkeeping regimes at macro level.

Independent recordkeeping authority with powers adequate to its purpose

ofessional standards and best practice promulgated and accepted by society.
ompliant recordkeeping systems at micro level.
eneficial alliances with other accountability players and relationships of trust with
ccountability stakeholders

Recordkeeping in this small office creates records supporting its daily business on the left side of the room (just out of the picture) and files its completed business records in the cabinets shown on the right, where they are still accessible, but out of the way. To safeguard their vital records and accommodate their statutory retention requirements, the managers utilise the longer term and security storage services of their public records repository.

Facts, figures and encouragements for good recordkeeping

Authority, reponsibility and powers

It is vital that the top level of decision making understand and approve the recordkeeping regime, its functions and powers; without such recognition, the regime will be unable to fulfil its organisation-wide responsibilities for ensuring evidence and will thus expose the organisation to unacceptable levels of managerial and regulatory risk. The need for AUTHORITY, RESPONSIBILITY and POWER can be articulated in the following way:

One of the most critical tools that a professional recordkeeper can possess is sufficient AUTHORITY to obtain compliance with his/her regime's policies and procedures. This authority to manage/control is granted by the ultimate decision-making entity within an organisation or bureaucratic system and is normally embodied in legislation(public sector) or in executive /administrative orders at the highest level (private sector).

The allocation of regime RESPONSIBILITY is twofold involving (i) Functional or operational responsibility to vest full responsibility and control over all aspects of records making, using and keeping within the organisation's entire scope of operations in the recordkeeping regime and (ii) Managerial authority which involves giving the recordkeeping professional the authority for developing/implementing/revising/enforcing requirements, standards and guidelines (policies, procedures) for all activities/resources which influence the quality and quantity of records throughout the entire management continuum of records making and keeping.

The POWER to carry out recordkeeping responsibility is embodied in regulations/administrative arrangements, standards, policies and guidelines issued by the recordkeeping regime acting on behalf of the highest authority. These documents specify what can and cannot be done across all activities which comprising and affecting the capture, preservation and accessibility of essential evidence as records. Ideally, all proposed actions impacting upon the quality or quantity of records media, file components, file housings, records creation, records storage areas, records creating/processing/storing technology, reprographics, records and files identification, documentation, maintenance and handling, records access & retrieval systems, automation, disposal, destruction, surveys, vital records protection, etc. should be subject to a coordinated approval and review process.

Unfortunately 1 and 2 in some settings are treated as if they were separate matters, rather than the two halves of the management whole. In such cases, an early and strong effort must be made to reunite and balance them; otherwise your regime cannot not be effective.

In addition to the powers Ketelaar recommends, there are other measures that can facilitate recordkeeping effectiveness - see Powers assisting recordkeeping effectiveness.

Counting and accounting: Communicating the 'value' of recordkeeping

One of the difficulties alluded to by many recordkeepers is communicating the value of recordkeeping to non-specialist stakeholders, particularly to those who pay the bills and expect results and value for money. The challenge is to measure the impact of recordkeeping activity BEFORE and AFTER an interval and then express the outcomes in terms that the target audience values.

Creating and caputuring the information embodied in records represents a considerable investment. One has only to look at the salaries of those responsible for preparing and analysing high level management documents and reports to see that these items have cost a bundle. In addition, there are those vital databases of client informmation, product inventories, design specifications and plans, If these were lost or could only be found with a great amount of time and effort, the resulting cost in terms of delay would be huge.

4.1.3.1.15 **Recordkeeping rewards**

It is no surprise that progressive public and private organisations are realising that ONLY a well-managed recordkeeping program or regime provides the full, accurate and trustworthy evidence needed for optimum rewards such as:

1. Management decision support
2. Compliance with legislative/regulatory requirements
3. Risk management, litigation protection and support
4. Organisational continuity, efficiency and productivity
5. Corporate knowledge base quality control and vital asset protection
6. Fountainhead of societal conscience and memory

4.1.3.1.16 **Recordkeeping and related professions**

While recordkeeping institutions are responsible for capturing and maintaining the documentary evidence important to society, they share the overall role of knowledge preservation and cultural transfer with other heritage management institutions including libraries and museums.

4.1.3.1.16.1 **What's the difference?**

Chart 1: The institutions and their holdings compares the traditional professional responsibilities of registries, archives, libraries and museums

Chart 2: Access to facilities and holdings compares the institutions differing levels of user access.

Chart 3: Professional staff and issues compares the qualifications and professional concerns of staff at registries, archives, libraries and museums.

Most people base their expectations of original sources and archival research on early experiences at school and their use of local public libraries.
Many are surprised that they cannot 'browse' the shelves and borrow originals of records for use at home. They soon understand the need to protect irreplaceable materials by using facsimile copies.

4.1.3.1.17 **Recordkeeping and specialist recordkeepers**

The activities of fully mature recordkeeping regimes document the present and reconstruct the past; they serve business and culture equally through work in offices and in repositories. Overall, the professionally educated recordkeeping specialists working in them are competent to perform the Duties of a recordkeeping specialist.
As we have seen almost every process people undertake in the world generates or involves some form of record and EVERYBODY, not just specialists, is involved. We are all de facto recordkeepers, though few of us may be aware of our role as such. However, in this segment, we concentrate on those who undertake the design and management of recordkeeping systems and service regimes as paid professional work. Recordkeeping specialists oversee the infrastructure - the principles, standards, policies, plans, guidelines and technologies - and provide the advice and support that enables people in different contexts to have the documentation required to meet their personal, business, regulatory and cultural obligations.

CHAPTER 5

ARCHIVE PROGRAM

When you click for open our program Fig 5.1 will open.than it will ask you username and password .You can see it in Fig 5.2

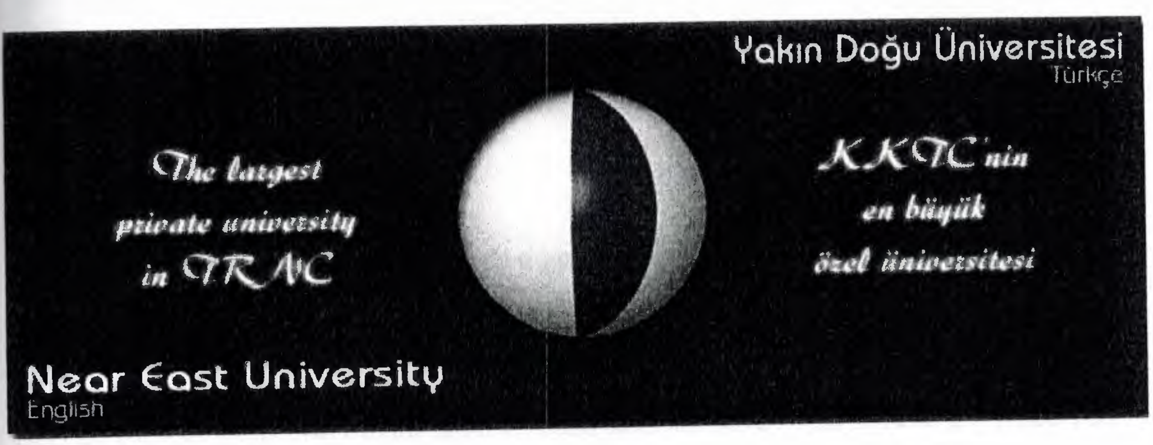
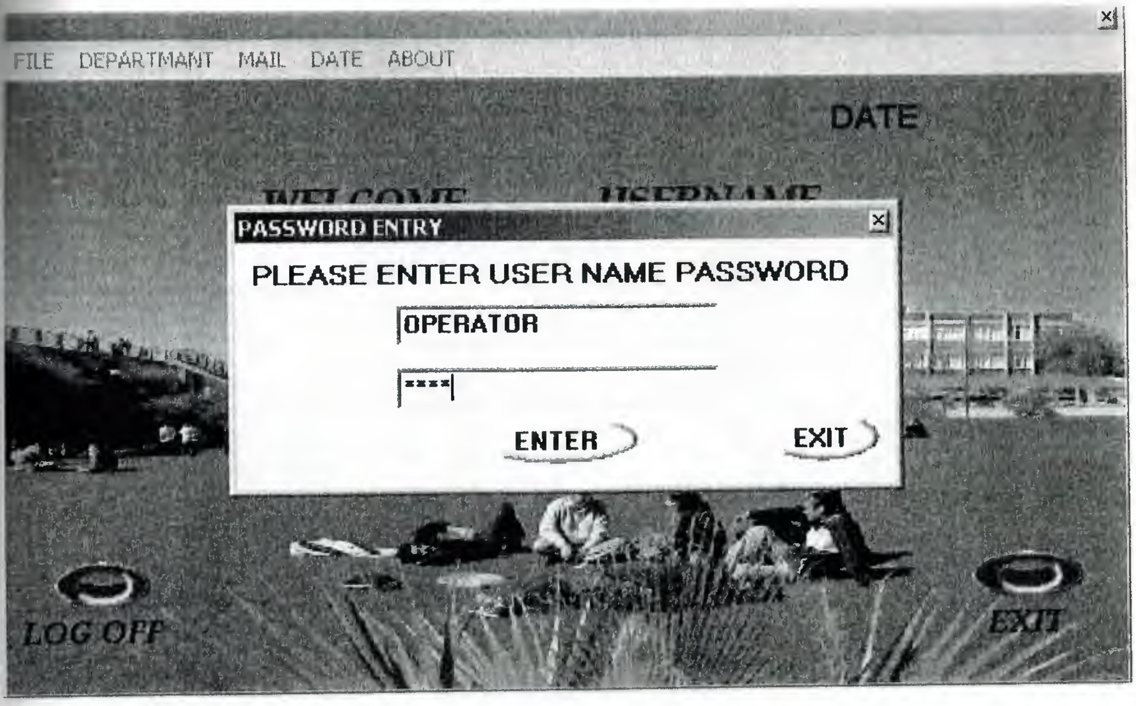

Yakın Doğu Üniversitesi
Türkçe
The largest private university in TRNC
KKTC'nin en büyük özel üniversitesi
Near East University
English

Fig 5.1

If you do not know username and password you can not enter this program.



PLEASE ENTER USER NAME PASSWORD

OPERATOR

****

Fig 5.2

In fig5.2 write username and password than click enter for use the archive program.

In our program we have File , Department , Mail , Date and About menus (shown in fig 5.3)

Lets go to start from File menus .In file menu also we have 5 parts these are Received File , Send Files , Reports , List and Close menus.

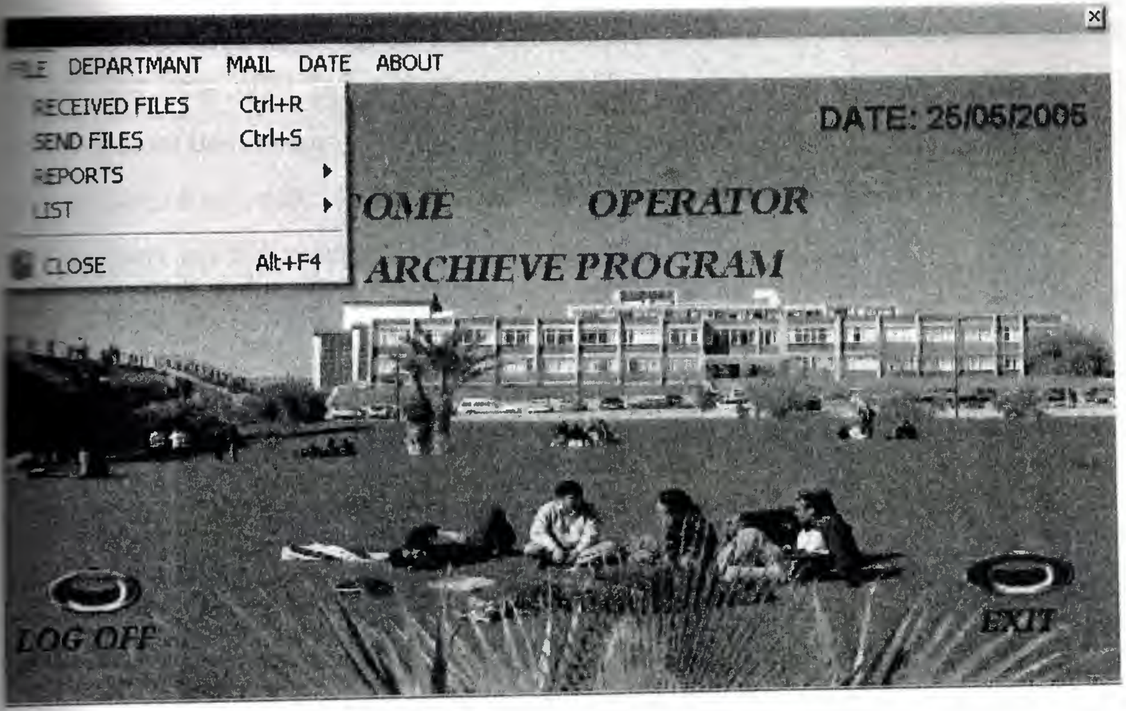
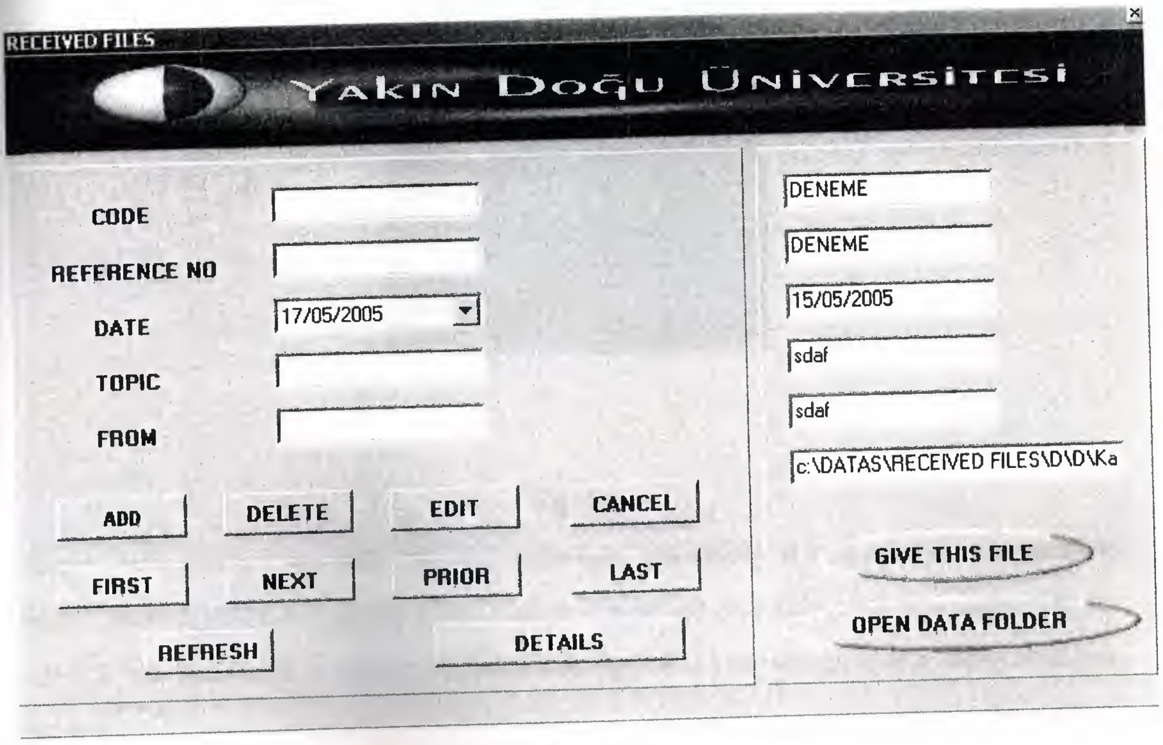


Fig5.3

For Received Files we have to click received files parts or we can use also for short way. We are using this part for working with received documents .Its shown in fig 5.4.

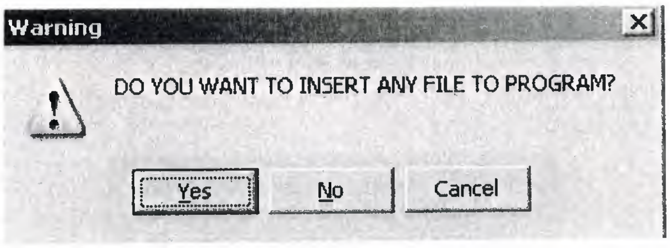


CODE	
REFERENCE NO	
DATE	17/05/2005
TOPIC	
FROM	

DENEME
DENEME
15/05/2005
sdaf
sdaf
c:\DATAS\RECEIVED FILES\D\D\Ka

Fig 5.4

As you see in fig 5.4 in this part we can add , delete and edit .We can see details and we can use first ,next ,prior ,last ,cancel and refresh button .you can understant what they are doing from their name .They are simple to understand. For add we have to write code , reference no , date , topic and from then click to add button than it will ask if we want to insert any file to program .Like Fig 5.5


Warning

DO YOU WANT TO INSERT ANY FILE TO PROGRAM?

Fig 5.5

If we want we have to click yes than again it will ask what we want to add word or scan. Shown In fig5.6.If it is word, word documents will open else scan documents will open. If we don't want to add any file we have to click no and save our work in the program.

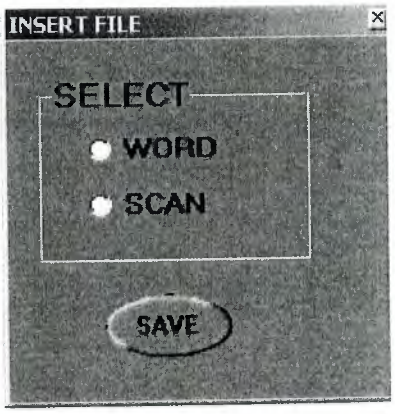
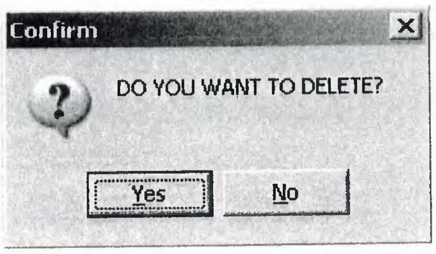


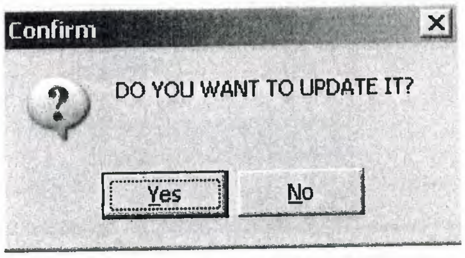
Fig 5.6

Delete button when we will click delete button for delete it will ask "do you want to delete" shown in fig 5.7. If yes it will delete it else not .For Edit button it same also it will ask "do you want to update it" shown in fig 5.8 if yes we can make some changes else not.


Confirm

DO YOU WANT TO DELETE?

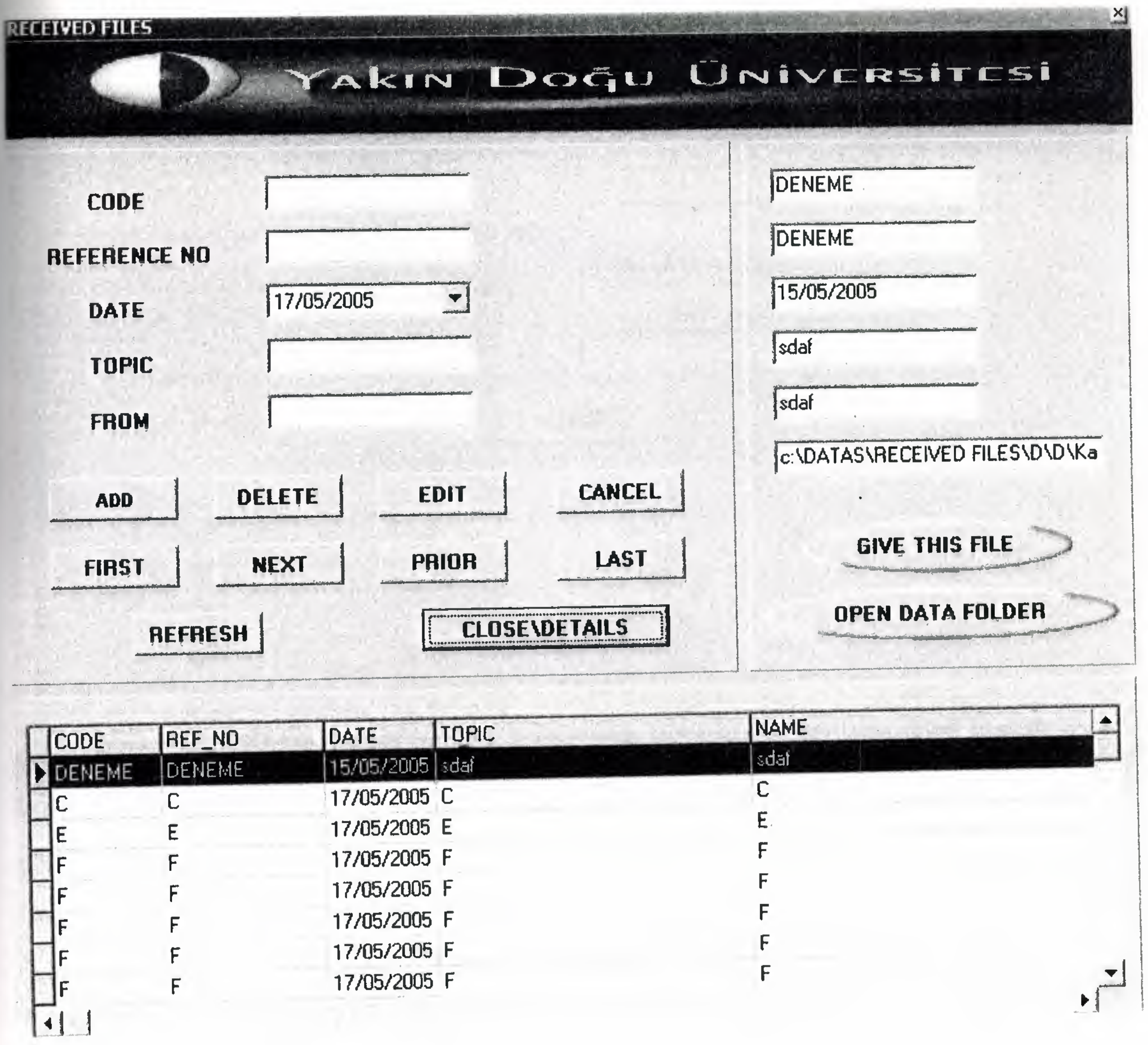
Fig 5.7


Confirm

DO YOU WANT TO UPDATE IT?

Fig 5.8

If we want to see details we have to click detail button. Shown in Fig 5.9 when we click it , it will show details then, button name will change its name to close/detail and if click close detail, button it will close the details and it will change button name to details again .



RECEIVED FILES

YAKIN DOĞU ÜNIVERSITESI

CODE	
REFERENCE NO	
DATE	17/05/2005
TOPIC	
FROM	

DENEME

DENEME

15/05/2005

sdaf

sdaf

c:\DATAS\RECEIVED FILES\D\D\Ka

CODE	REF_NO	DATE	TOPIC	NAME
DENEME	DENEME	15/05/2005	sdaf	sdaf
C	C	17/05/2005	C	C
E	E	17/05/2005	E	E
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F

Fig 5.9

If we want to send this document to other personel we have to use give this file button. Then it will open new page its shown in Fig 5.10 we will find which one we want to send then it will and send it.

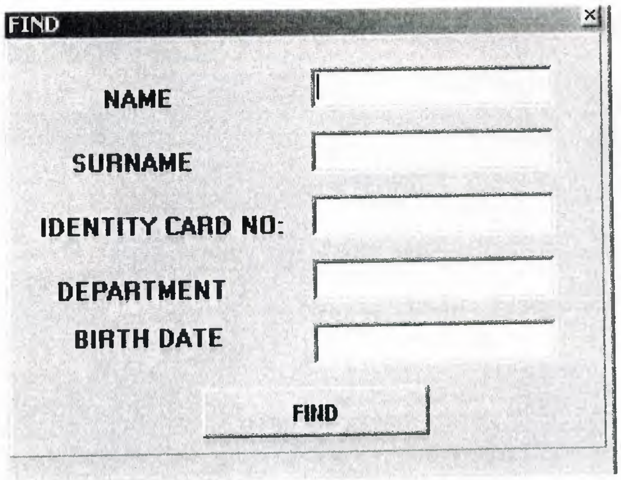


Fig 5.10

About open data folder button, which documents we saved it we can open it with this button .When we click it Fig 5.11 will open. Than we can see the document's .Also it can be scan pages here.

Fig 5.11

For Send Files we have to click Send Files parts or we can use also for short way. We are using this part for working with send document's .Its shown in fig5.12

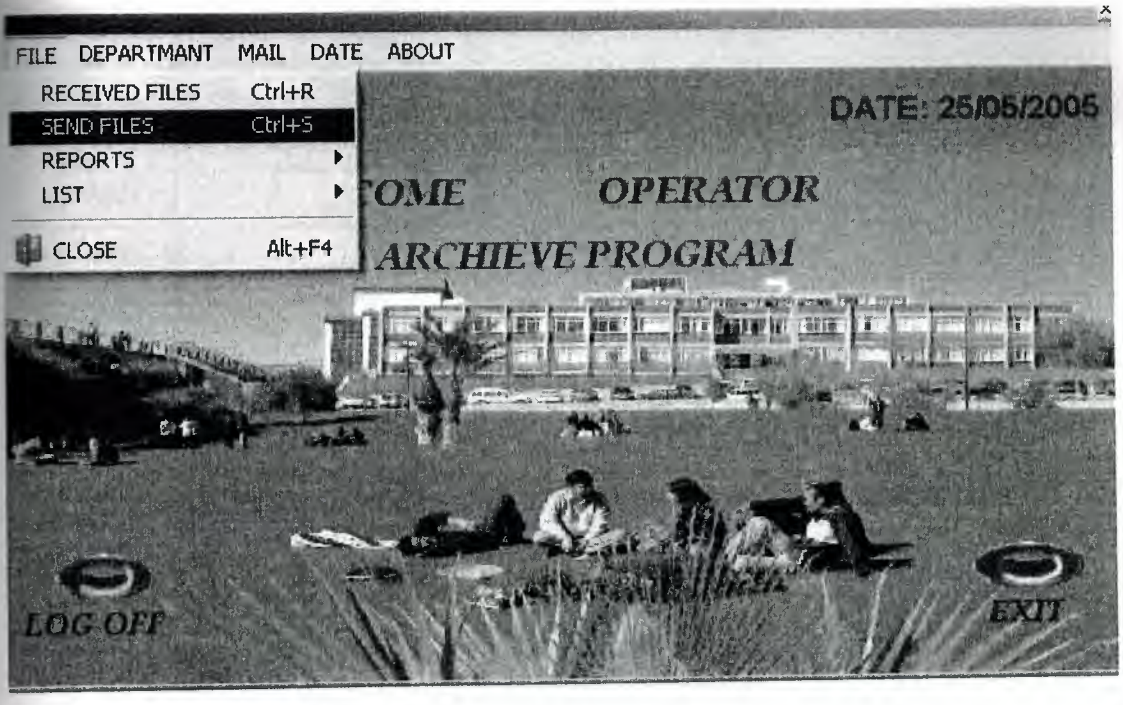
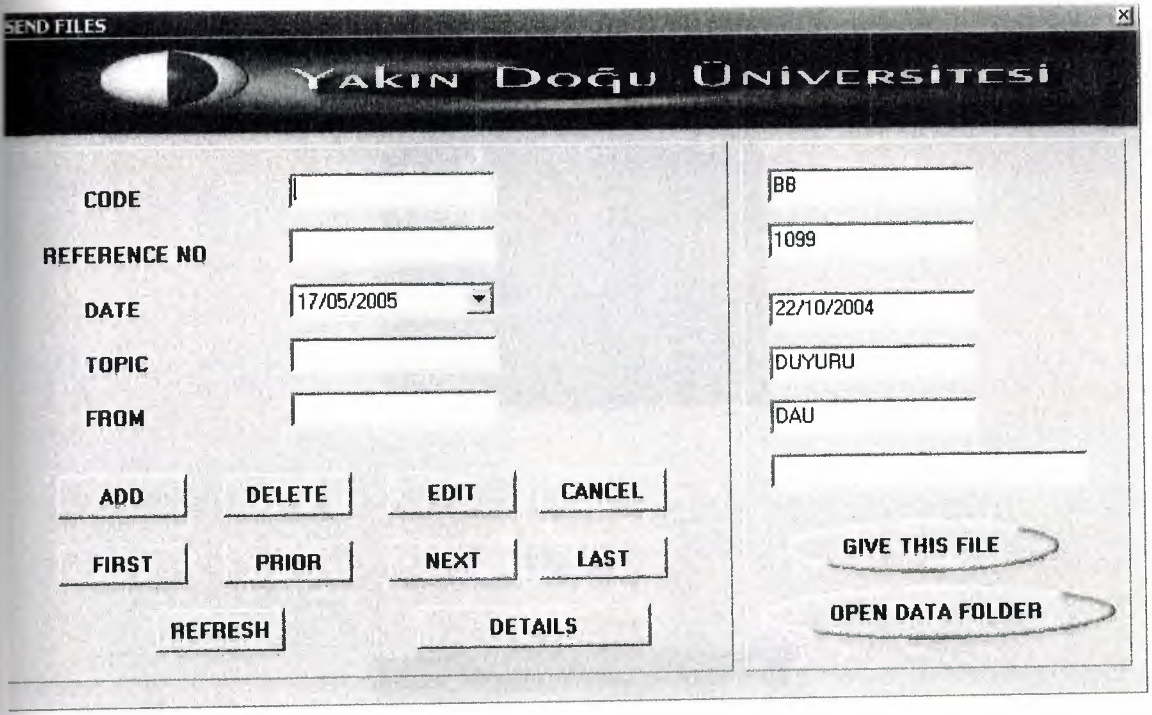


Fig 5.12

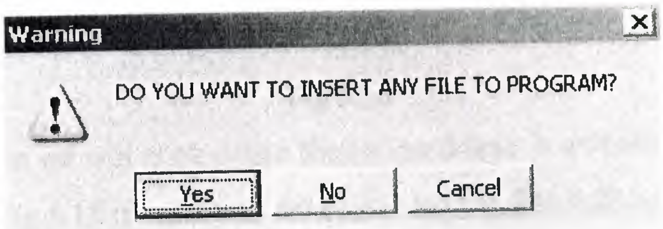
As you see in fig 5.13 in this part we can add , delete and edit .We can see details and we can use first ,next ,prior ,last ,cancel and refresh button .You can understant what they are doing from their name .They are simple to understand. For add we have to write code , reference no , date , topic and from then click to add button than it will ask if we want to insert any file to program. Like Fig 5.14



CODE
REFERENCE NO
DATE 17/05/2005
TOPIC
FROM

BB
1099
22/10/2004
DUYURU
DAU

Fig 5.13



DO YOU WANT TO INSERT ANY FILE TO PROGRAM?

Fig 5.14

As you see in Fig 5.14. If we want we have to click yes than again it will ask what we want to add word or scan. Shown In Fig 5.15. If it is word, word documents will open else scan documents will open. If we don't want to add any file we have to click no and save our work in the program.

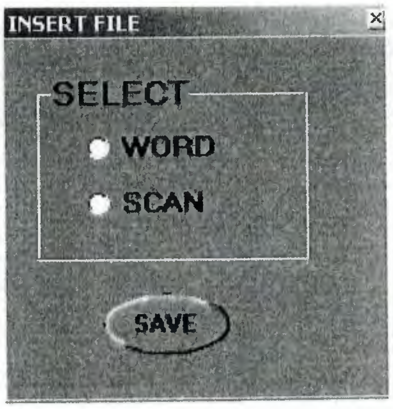
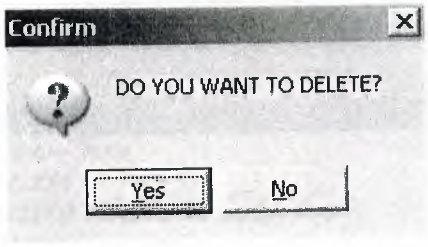


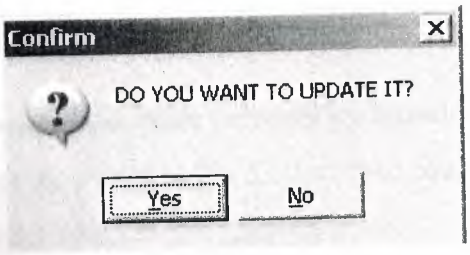
Fig 5.15



DO YOU WANT TO DELETE?

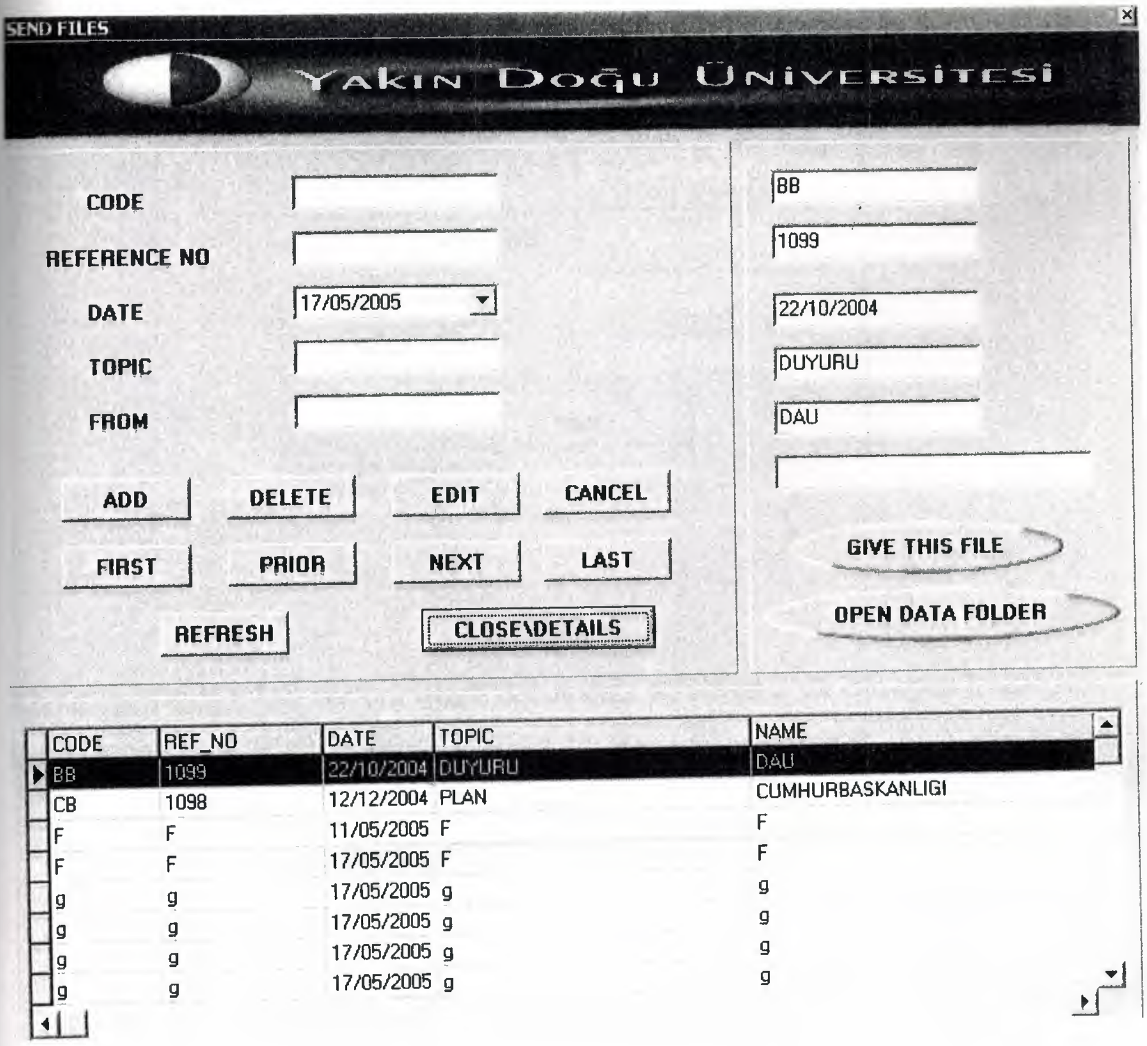
Fig 5.16

Delete button when we will click delete button for delete it will ask "do you want to delete" shown in Fig 5.15 if yes it will delete it , else not .For Edit button it same also it will ask "do you want to update it" shown in fig 5.16 if yes we can make some changes else not.



DO YOU WANT TO UPDATE IT?

Fig5.17



CODE	
REFERENCE NO	
DATE	17/05/2005
TOPIC	
FROM	

BB
1099
22/10/2004
DUYURU
DAU

CODE	REF_NO	DATE	TOPIC	NAME
BB	1099	22/10/2004	DUYURU	DAU
CB	1098	12/12/2004	PLAN	CUMHURBASKANLIGI
F	F	11/05/2005	F	F
F	F	17/05/2005	F	F
g	g	17/05/2005	g	g
g	g	17/05/2005	g	g
g	g	17/05/2005	g	g
g	g	17/05/2005	g	g

Fig5.18

If we want to see details we have to click detail button. Shown in Fig 5.18 when we click it , it will show details then, button name will change its name to close/detail and if click close detail, button it will close the details and it will change button name to details again .

If we want to send this document to other personel we have to use give this file button. Then it will open new page its shown in Fig 5.19 we will find which one we want to send then it will and send it.

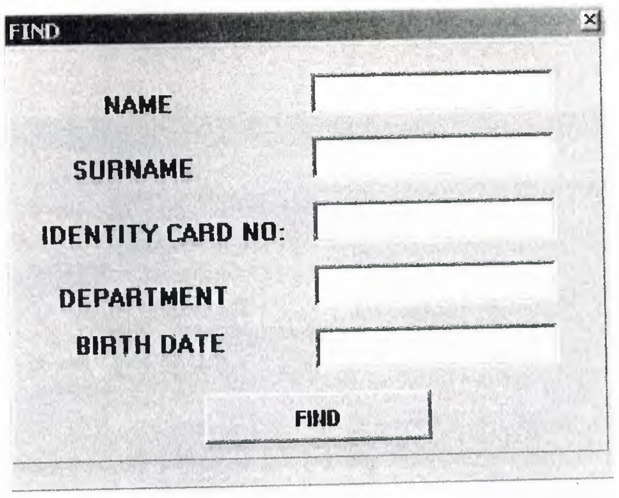


Fig 5.19

About open data folder button, which documents we saved it we can open it with this button.When we click it .It shown in Fig 5.20 than you can open it.

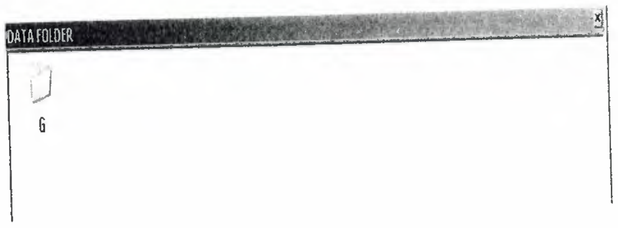
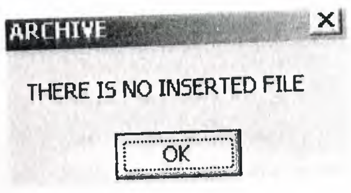


Fig 5.20

Than we can see the document's. Also it can be scan pages here ,but if you haven't any documents saved there it will give us attention shown in Fig 5.21.



THERE IS NO INSERTED FILE

Fig 5.21

In the reports we can two more part one for received file and one for send files .It shown in fig 5.21.

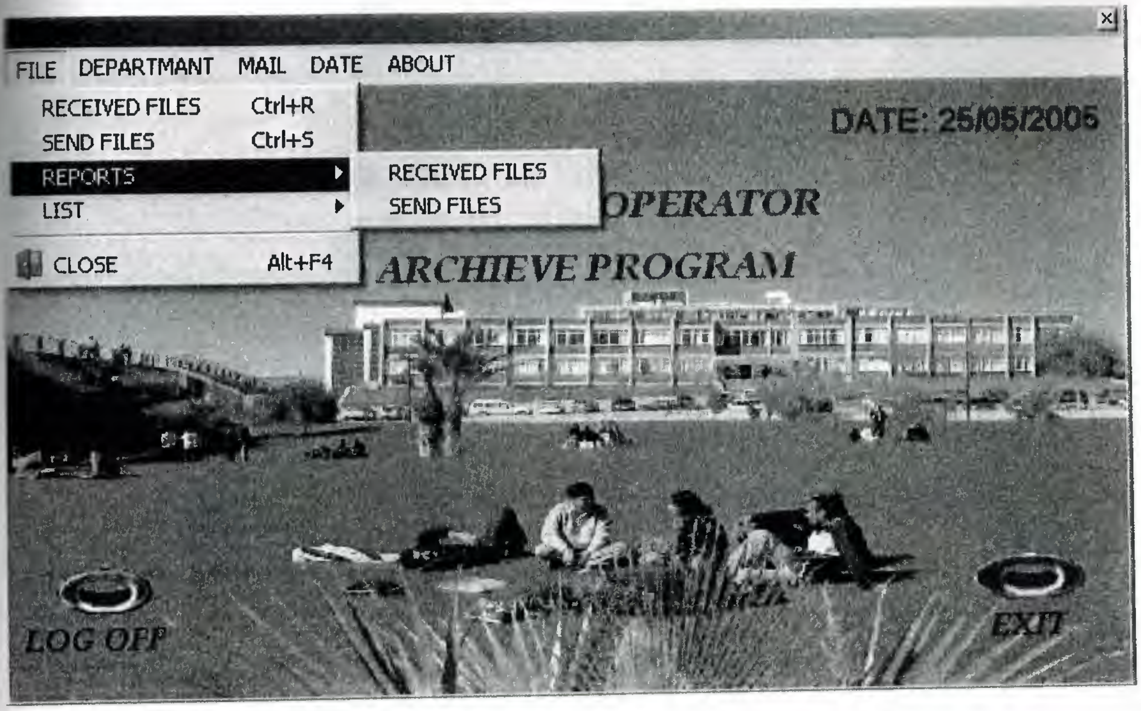


Fig 5.22

When you click received file , its shown in Fig 5.23 we see report for received files .Its shown in fig 5.24.

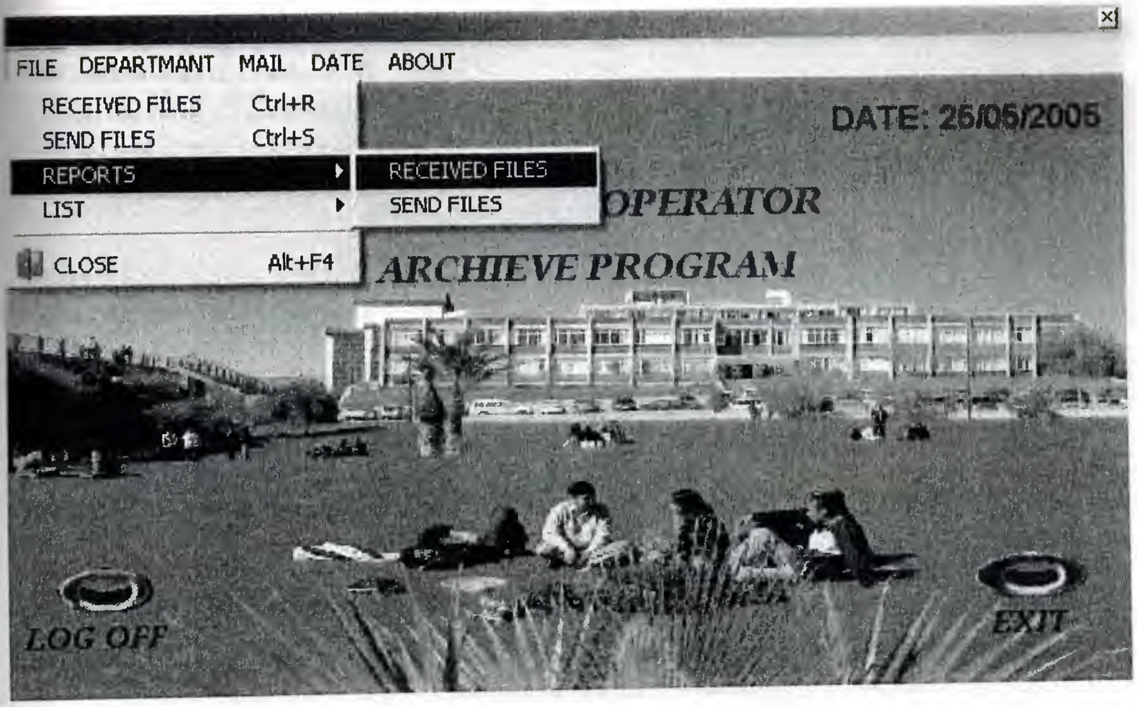
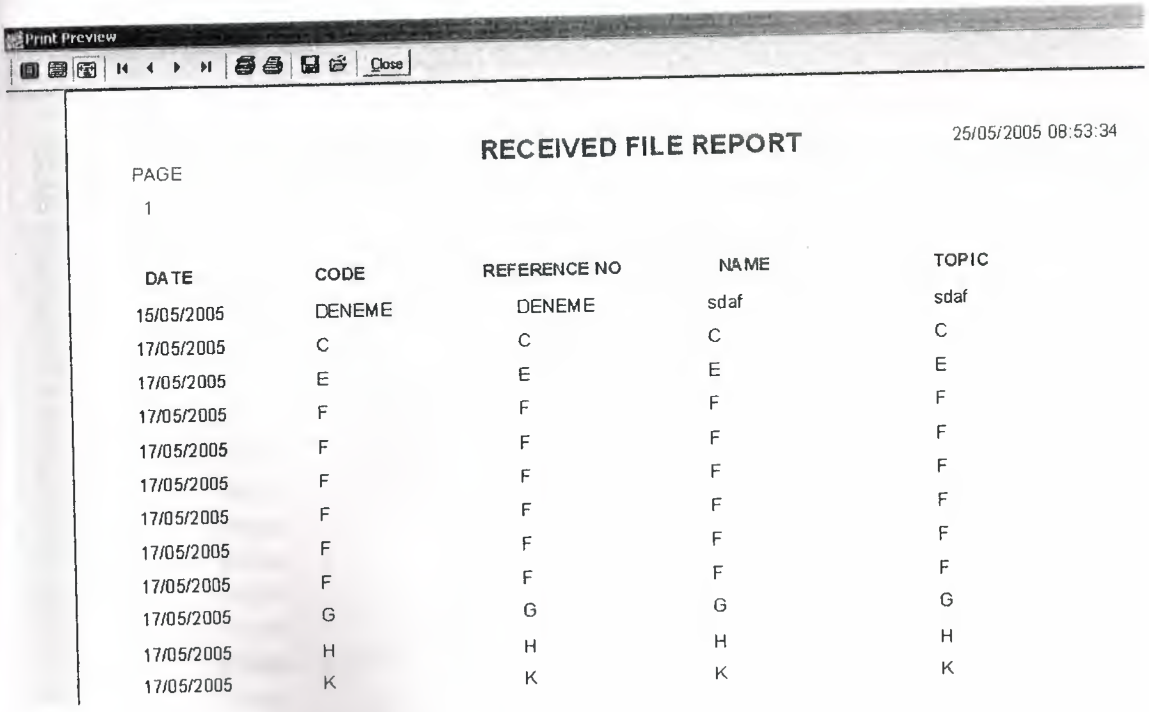


Fig 5.23



RECEIVED FILE REPORT

25/05/2005 08:53:34

PAGE
1

DATE	CODE	REFERENCE NO	NAME	TOPIC
15/05/2005	DENEME	DENEME	sdaf	sdaf
17/05/2005	C	C	C	C
17/05/2005	E	E	E	E
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	G	G	G	G
17/05/2005	H	H	H	H
17/05/2005	K	K	K	K

Fig 5.24

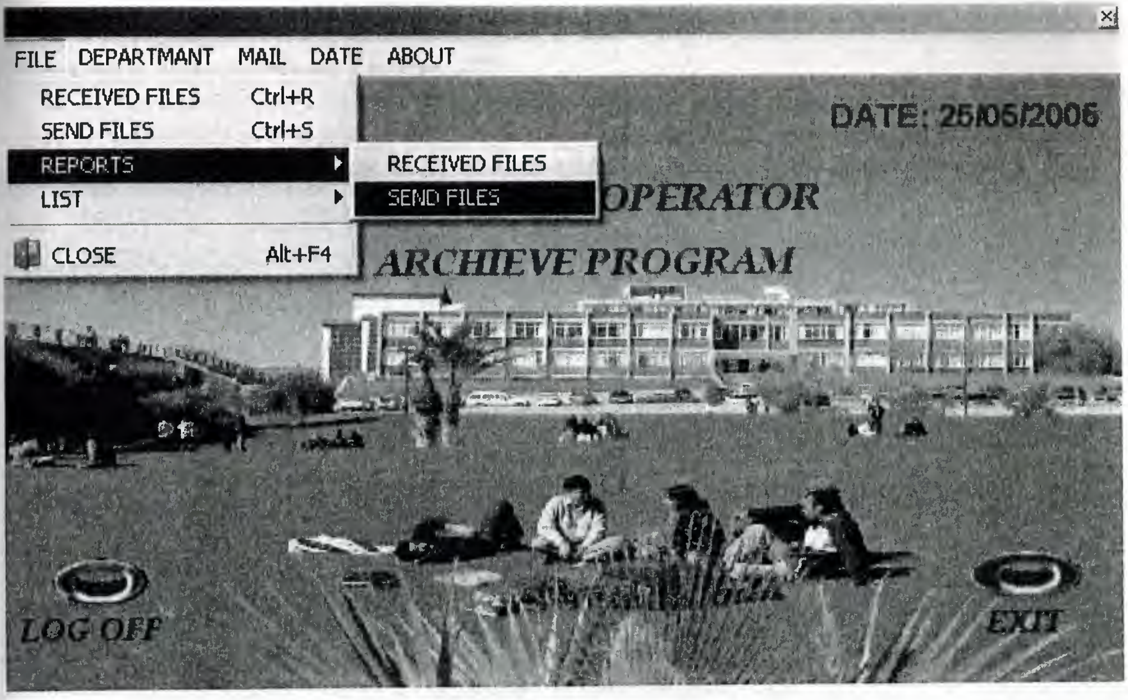
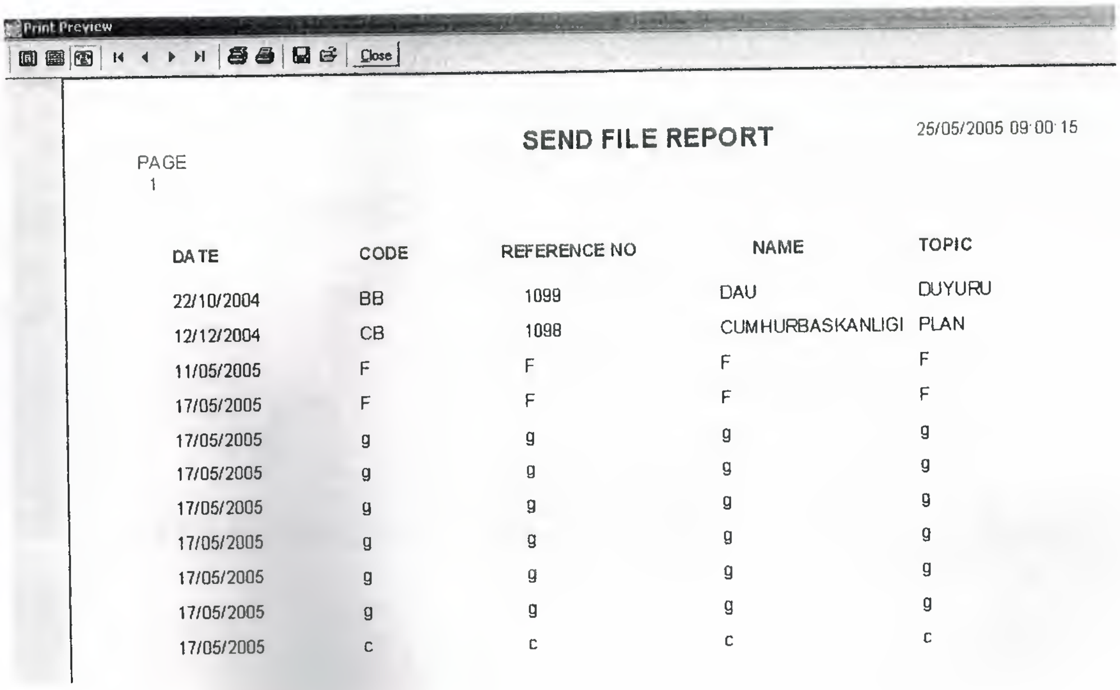


Fig 5.25

If you click send files in reports ,its shown in Fig 5.25 you will see the reports for send files .Shown in Fig 5.26.



SEND FILE REPORT

25/05/2005 09:00:15

PAGE
1

DATE	CODE	REFERENCE NO	NAME	TOPIC
22/10/2004	BB	1099	DAU	DUYURU
12/12/2004	CB	1098	CUMHURBASKANLIGI	PLAN
11/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	c	c	c	c

Fig 5.26

List part its look like reports we can also two parts in this section. These are received file and send files. It's shown in fig 5.27.

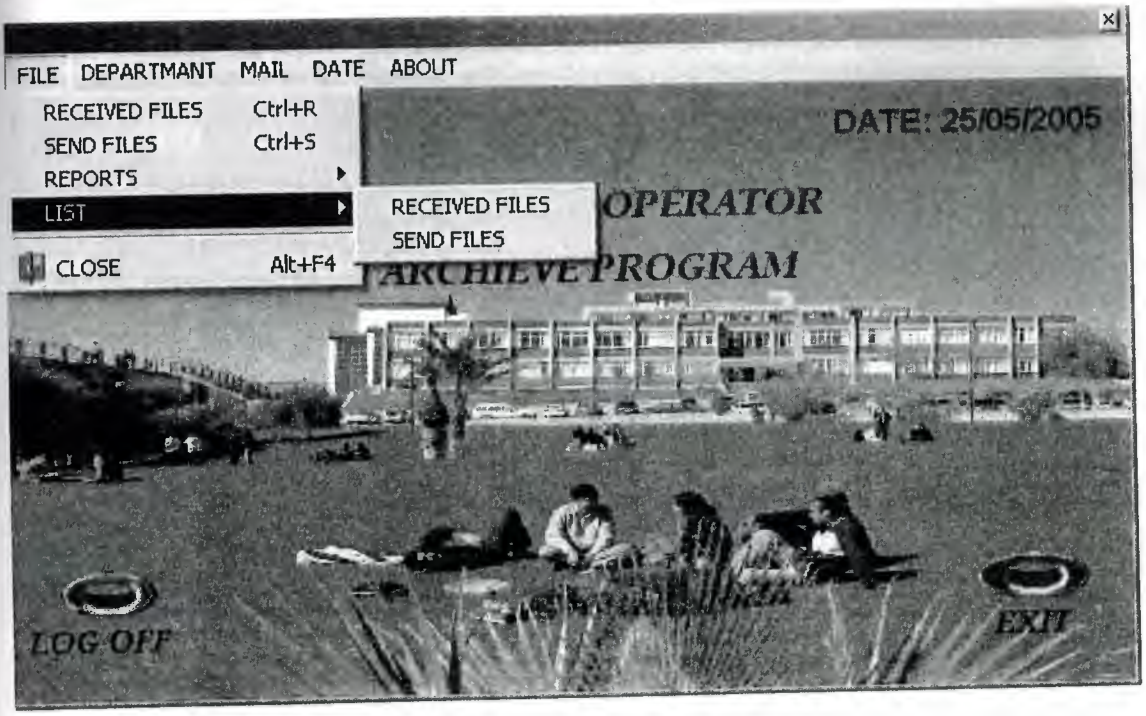


Fig 5.27

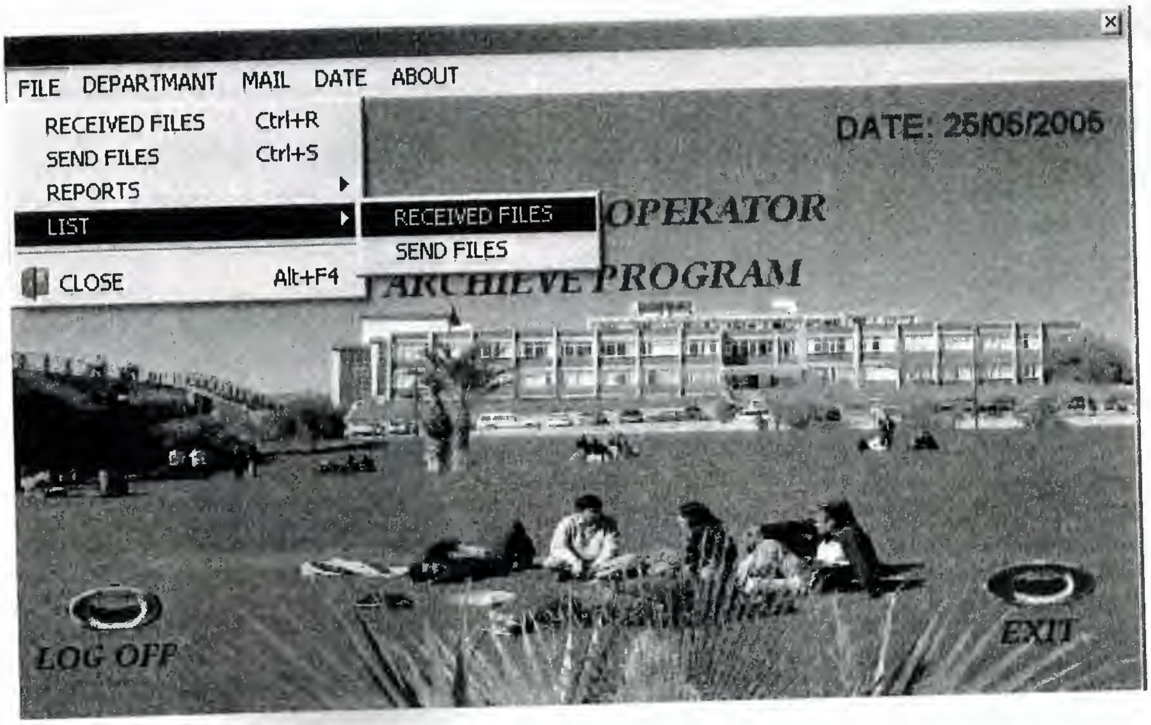


Fig 5.28

If we want see received files we have to click received files in Fig 5.28.When we click it new page will open it its shown in Fig 5.29.

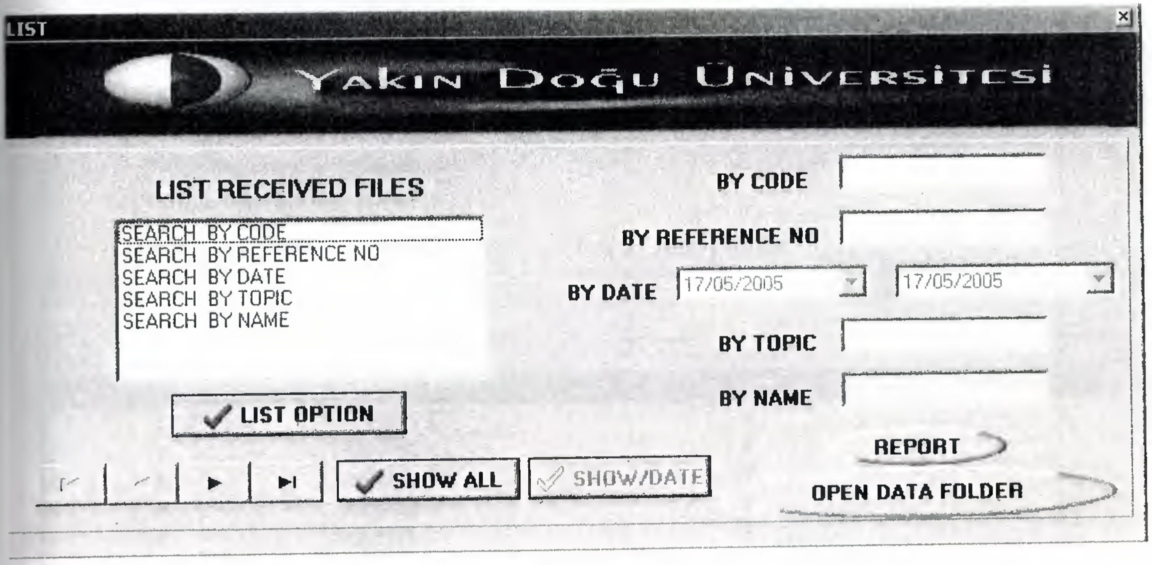


Fig 5.29

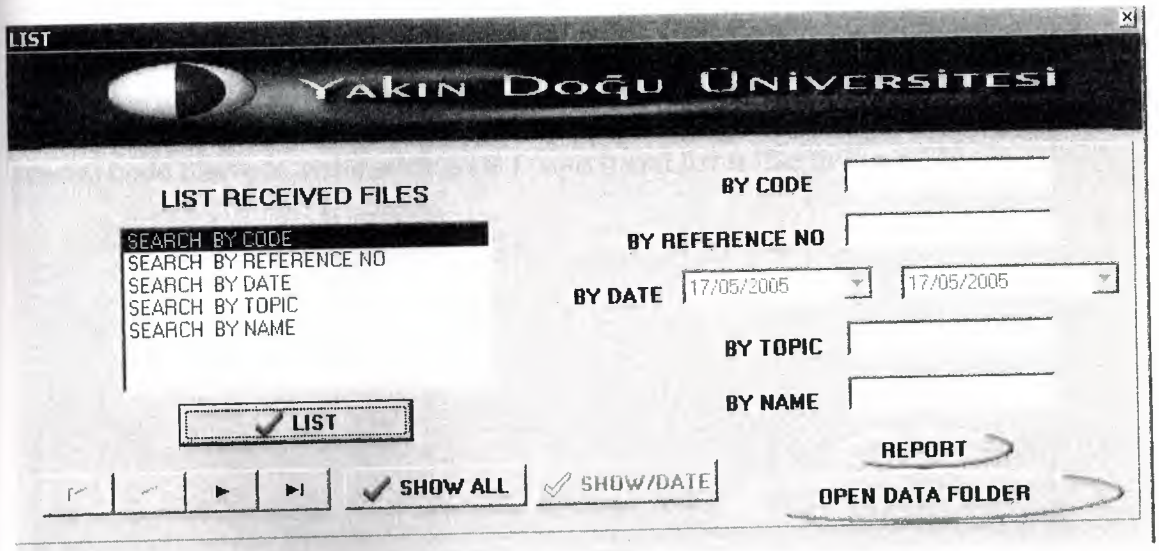
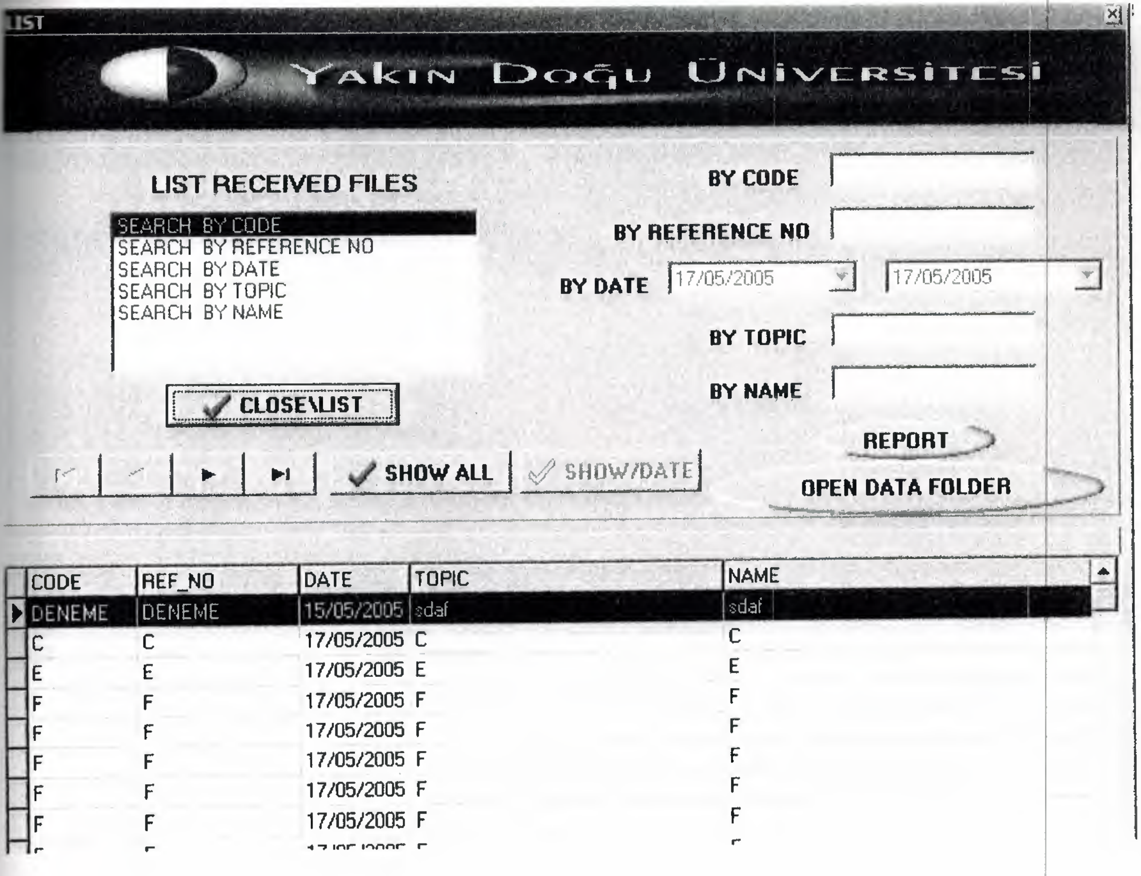


Fig 5.30

You have to choose which part you want to see its shown in Fig 5.31 and click list when you click it will show you details than list button will change its name to close\list and if you want to close details you can close it with use this button.

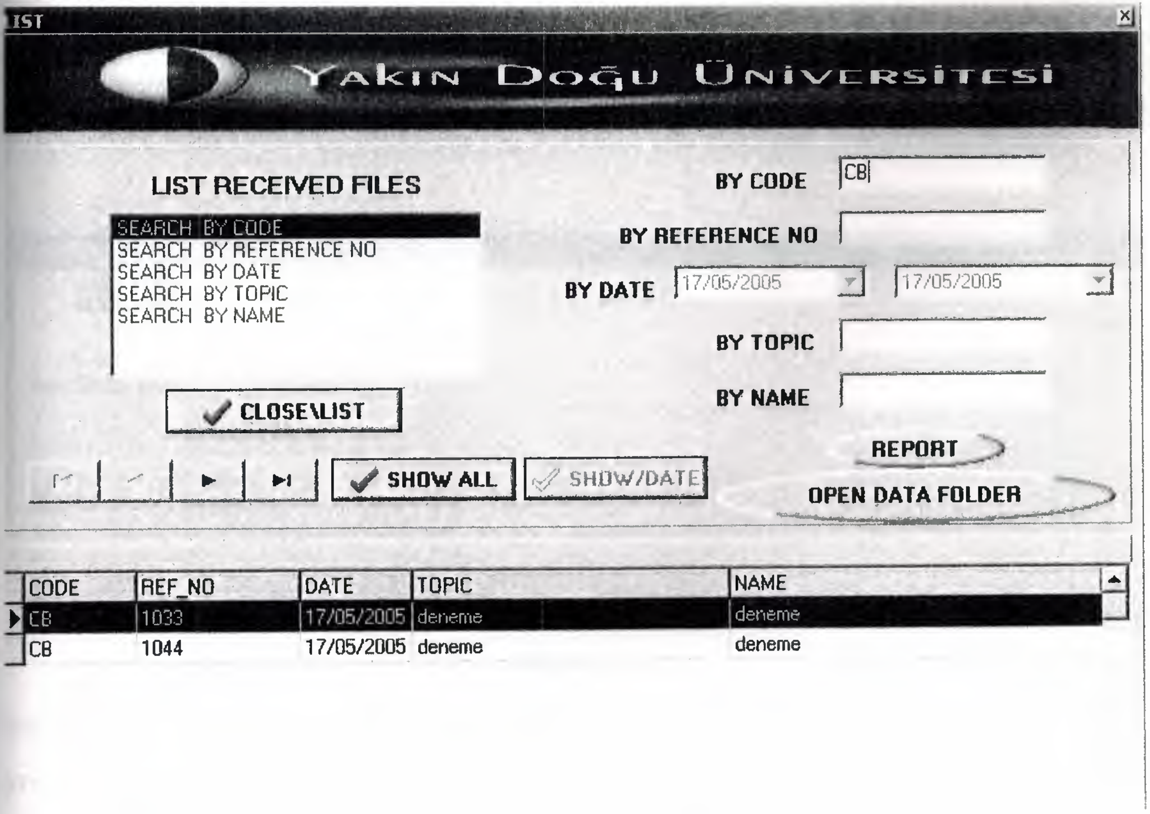


BY DATE 17/05/2005 17/05/2005

CODE	REF_NO	DATE	TOPIC	NAME
DENEME	DENEME	15/05/2005	sdaf	sdaf
C	C	17/05/2005	C	C
E	E	17/05/2005	E	E
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F

Fig 5.31

When you click which part you want for example I choose for search by code for special code I have to write wick code I want it and list it like in Fig 5.32.

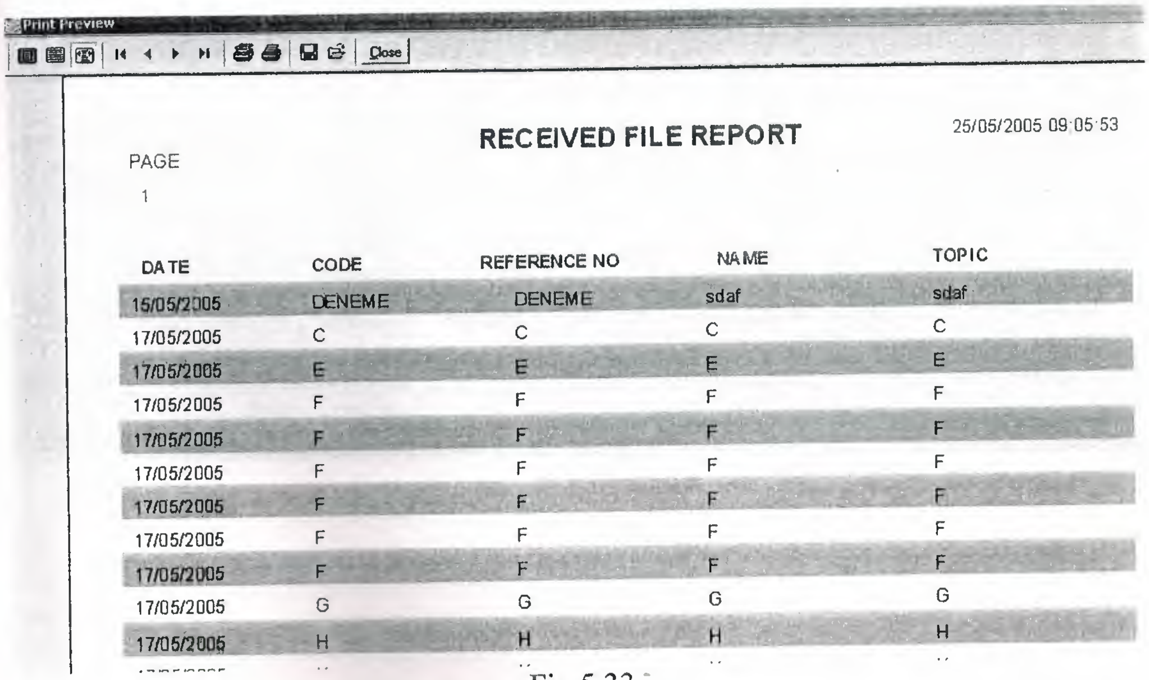


BY CODE: CB
BY REFERENCE NO:
BY DATE: 17/05/2005 17/05/2005
BY TOPIC:
BY NAME:

CODE	REF_NO	DATE	TOPIC	NAME
CB	1033	17/05/2005	deneme	deneme
CB	1044	17/05/2005	deneme	deneme

Fig 5.32

When you click reports button in Fig 5.32 you can see the reports in Fig 5.33.



RECEIVED FILE REPORT

25/05/2005 09:05:53

PAGE
1

DATE	CODE	REFERENCE NO	NAME	TOPIC
15/05/2005	DENEME	DENEME	sdaf	sdaf
17/05/2005	C	C	C	C
17/05/2005	E	E	E	E
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	G	G	G	G
17/05/2005	H	H	H	H

Fig 5.33

If you have some folder to see it in Fig 5.31 when you click the open data folder button you will see your folder its shown in Fig 5.34 and you can open it with double click.

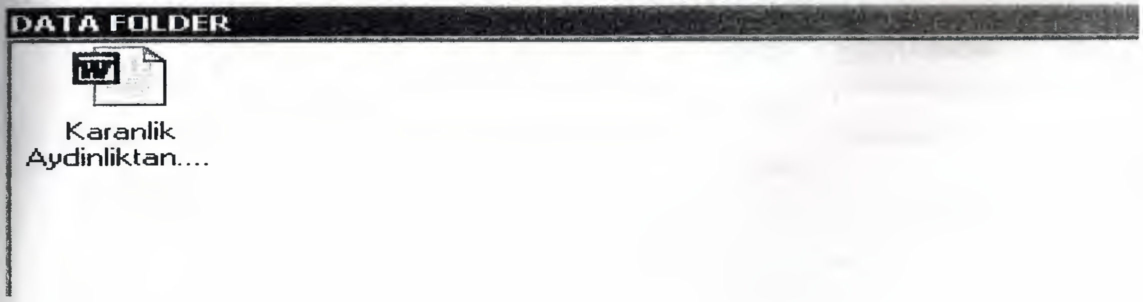


Fig 5.34

For see the send files in list part you have to click send files its shown in Fig 5.35.when you click it , it will open new page for send files its shown in Fig 5.36.

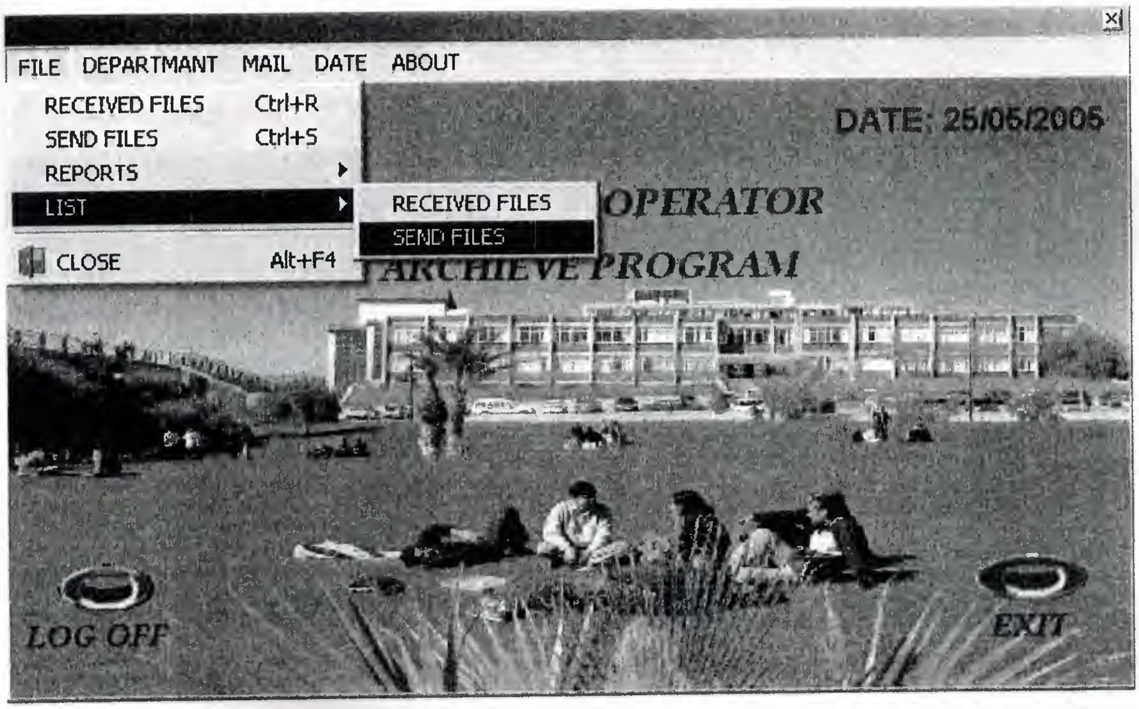


Fig 5.35

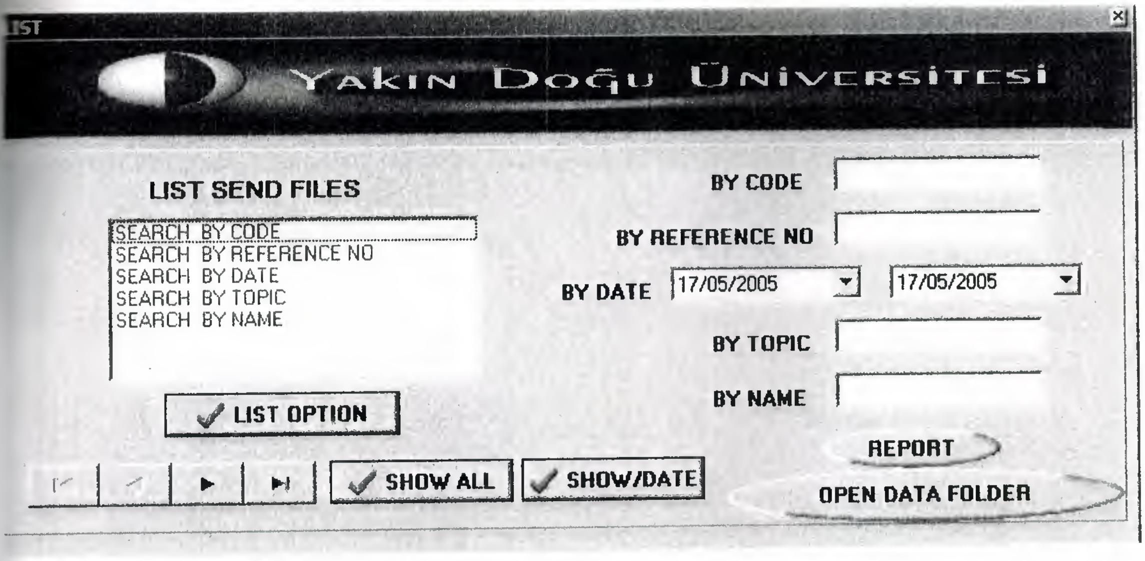


Fig 5.36

Its look like received file you have to choose which part you want to see its shown in Fig 5.37 and click list when you click it will show you details than list button will change its name to close\list and if you want to close detail you can close it with use this button .Its shown in Fig 5.38

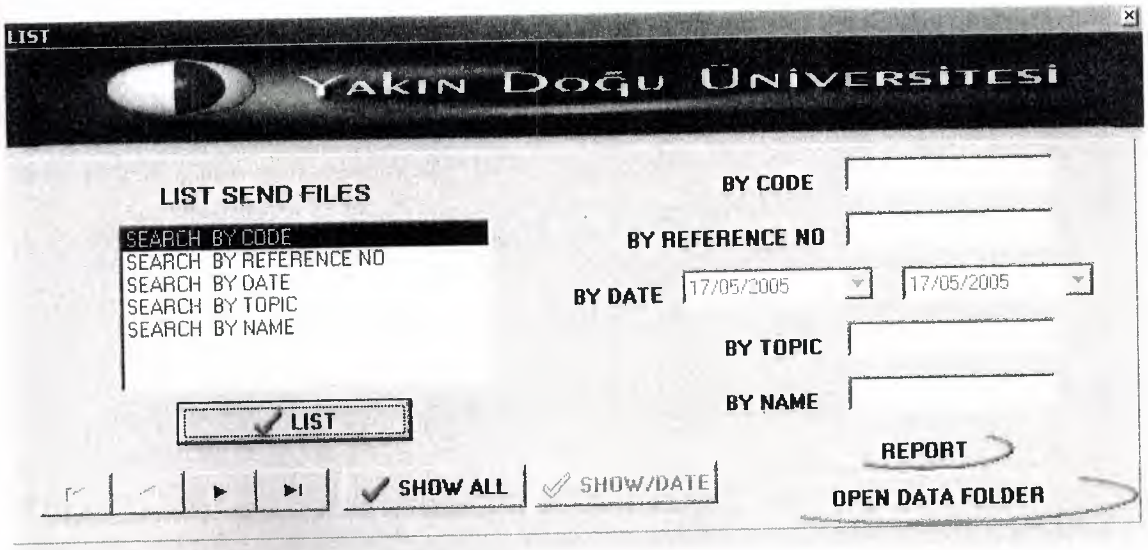
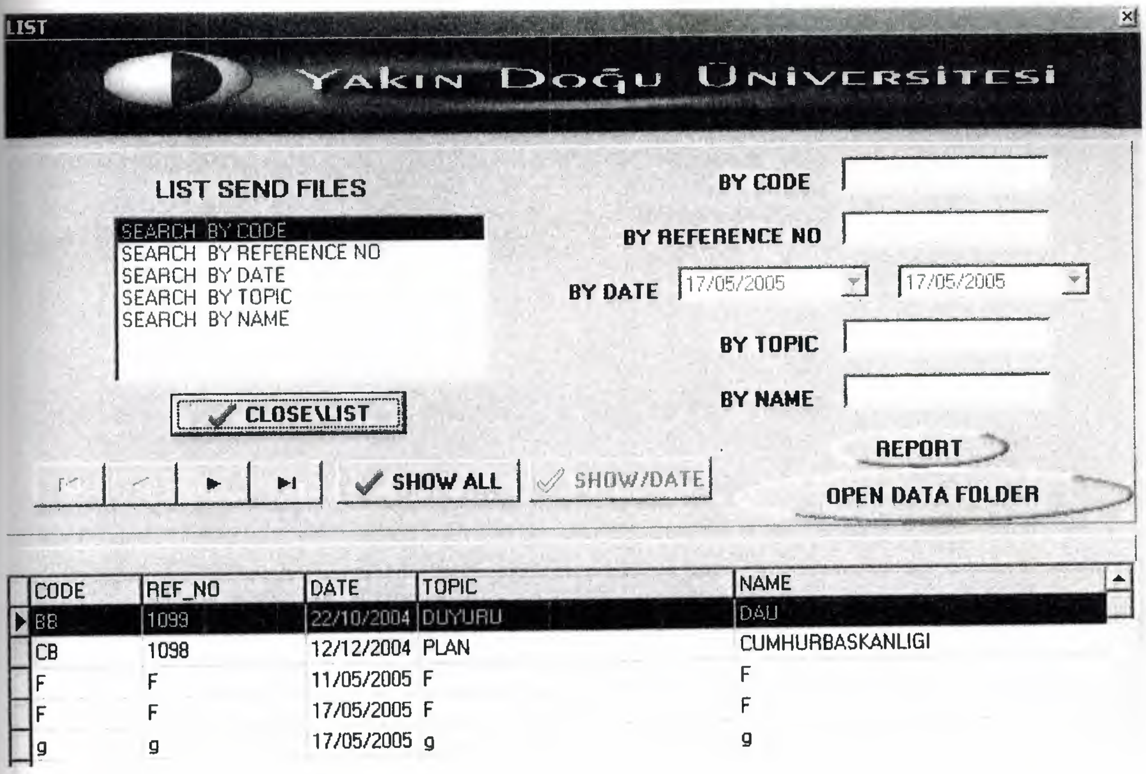


Fig 5.37

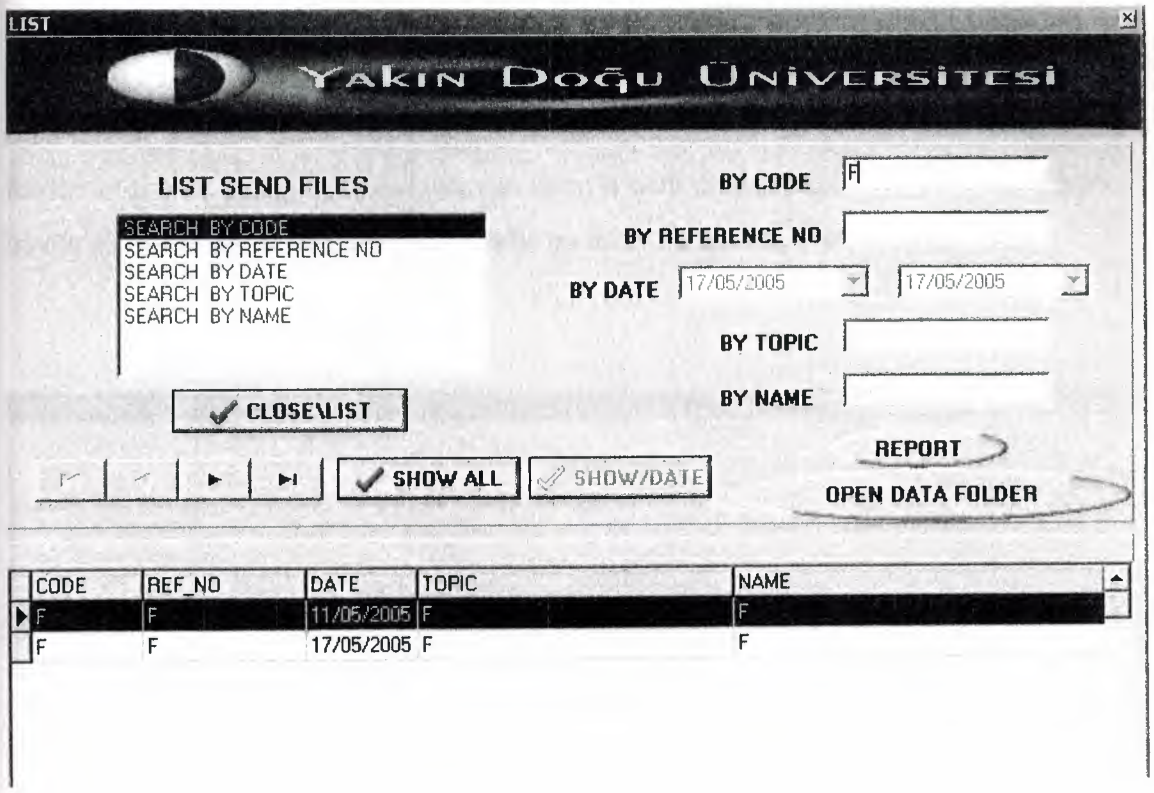


BY DATE 17/05/2005 17/05/2005

CODE	REF_NO	DATE	TOPIC	NAME
BB	1099	22/10/2004	DUYURU	DAU
CB	1098	12/12/2004	PLAN	CUMHURBASKANLIGI
F	F	11/05/2005	F	F
F	F	17/05/2005	F	F
g	g	17/05/2005	g	g

Fig 5.38

When you click which part you want for example I choosed for search by code for special code I have to write wich code I want it and list it like in fig 5.39

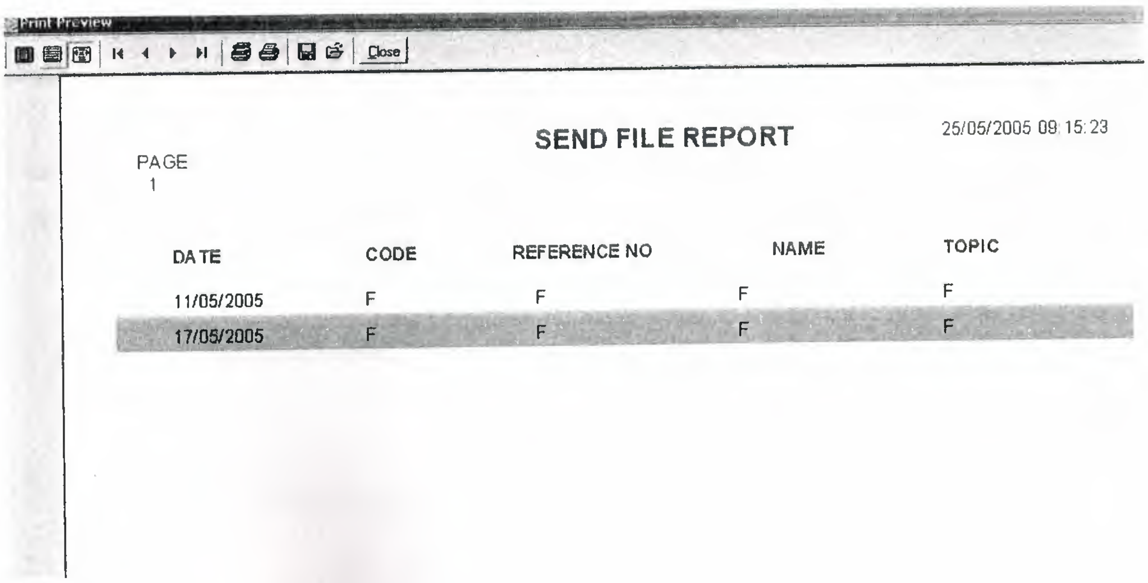


BY CODE: F
BY DATE: 17/05/2005 17/05/2005

CODE	REF_NO	DATE	TOPIC	NAME
F	F	11/05/2005	F	F
F	F	17/05/2005	F	F

Fig 5.39.

When you click reports button in Fig 5.37 you can see the reports in Fig 5.40.



SEND FILE REPORT

25/05/2005 09:15:23

PAGE
1

DATE	CODE	REFERENCE NO	NAME	TOPIC
11/05/2005	F	F	F	F
17/05/2005	F	F	F	F

Fig 5.40

If you have some folder which line you select it when you click the open data folder button it shown in Fig5.37 when you click the open data folder button you will see your folder its shown in Fig 5.41 and you can open it with double click . Else I mean if you haven't any folder it will tell you there is no inserted file its shown in Fig 5.42.

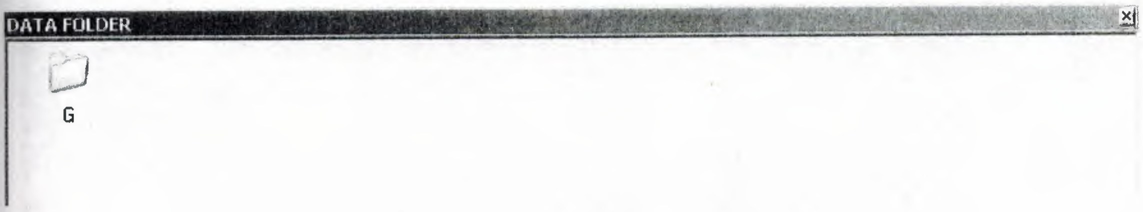
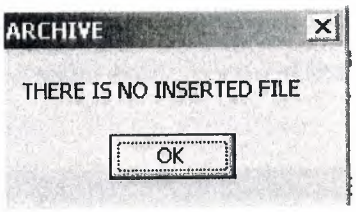


Fig 5.41



THERE IS NO INSERTED FILE

Fig 5.42

Last part for file menus is close its shown in fig 5.43 when you will chose this part you will close the archive program.

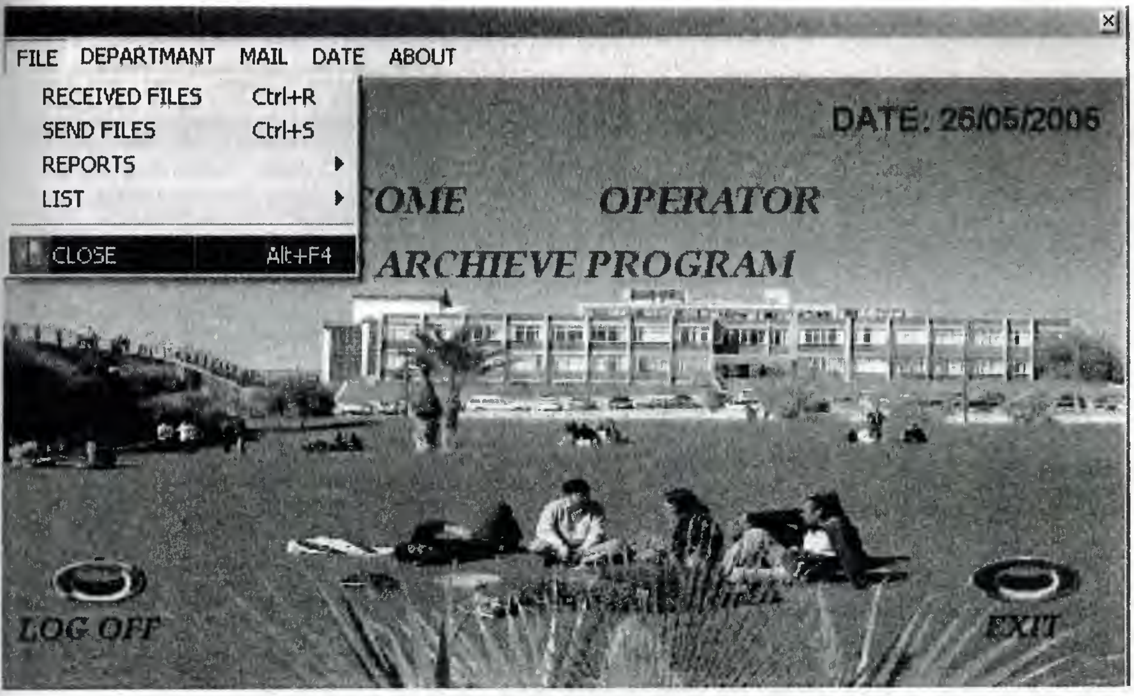


Fig 5.43

Second main menus for our program are department. Department menus have a four part. These are user settings, personal entry, personal list and given files list. As you can see the Fig 5.44.

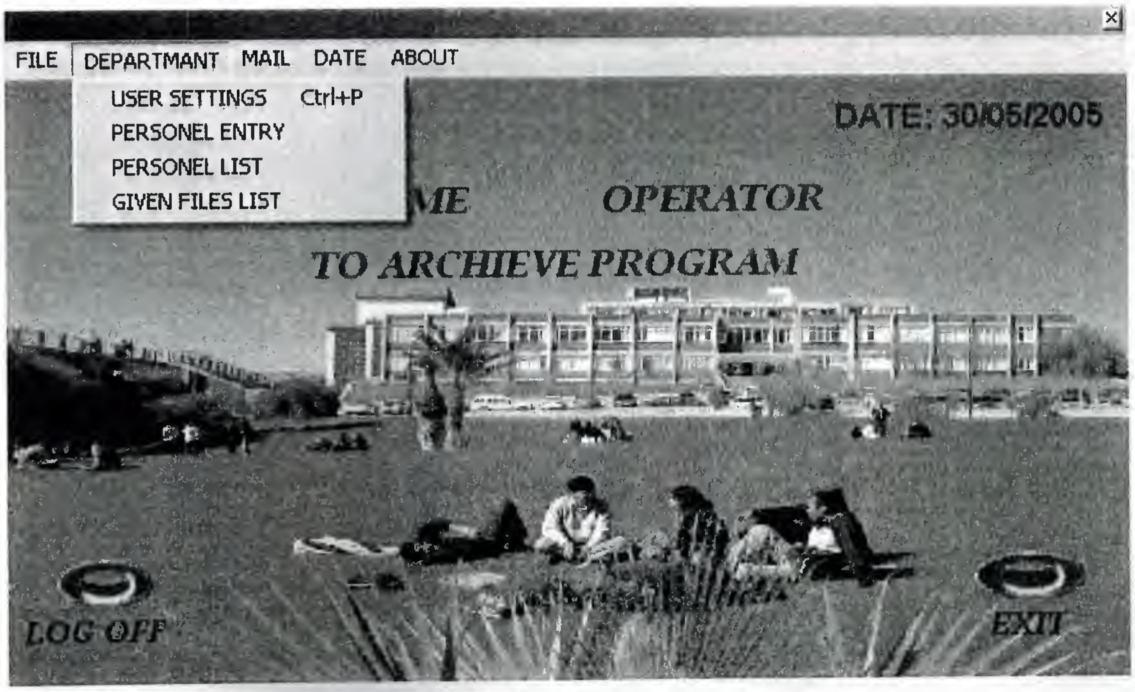


Fig 5.44

for enter the user settings we have to click user settings or we can also use short way to enter this part. Shown in Fig 5.45

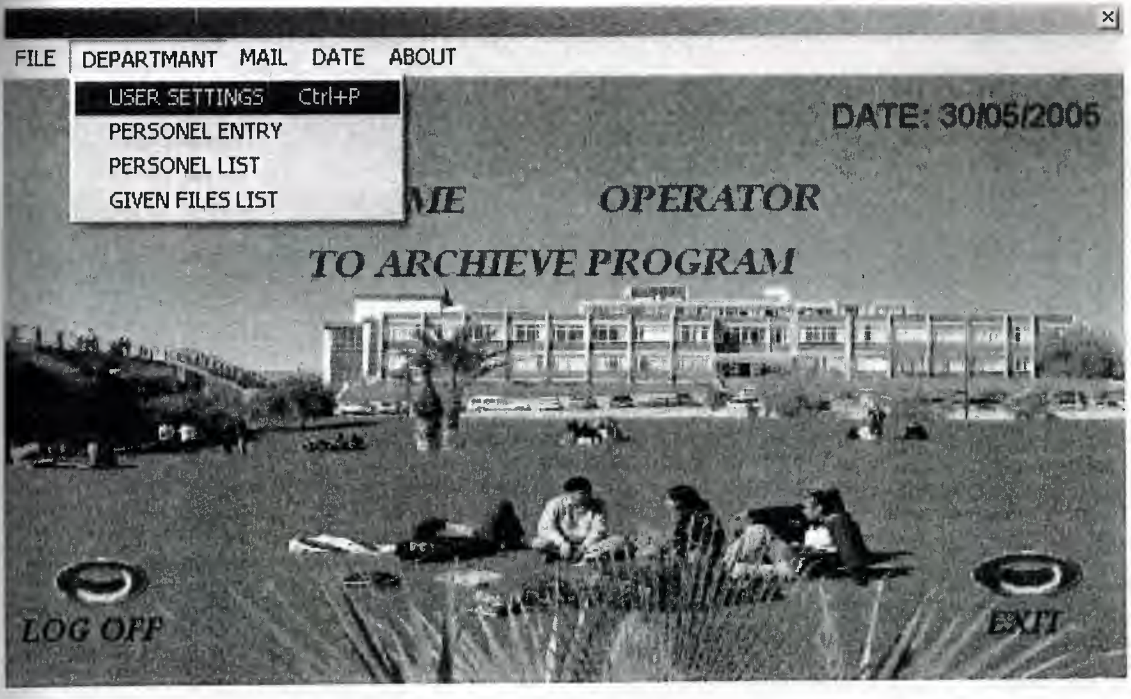


Fig 5.45

When we clicked user settings part it will open new page and it ill ask username and password for enter it shown in fig 5.46

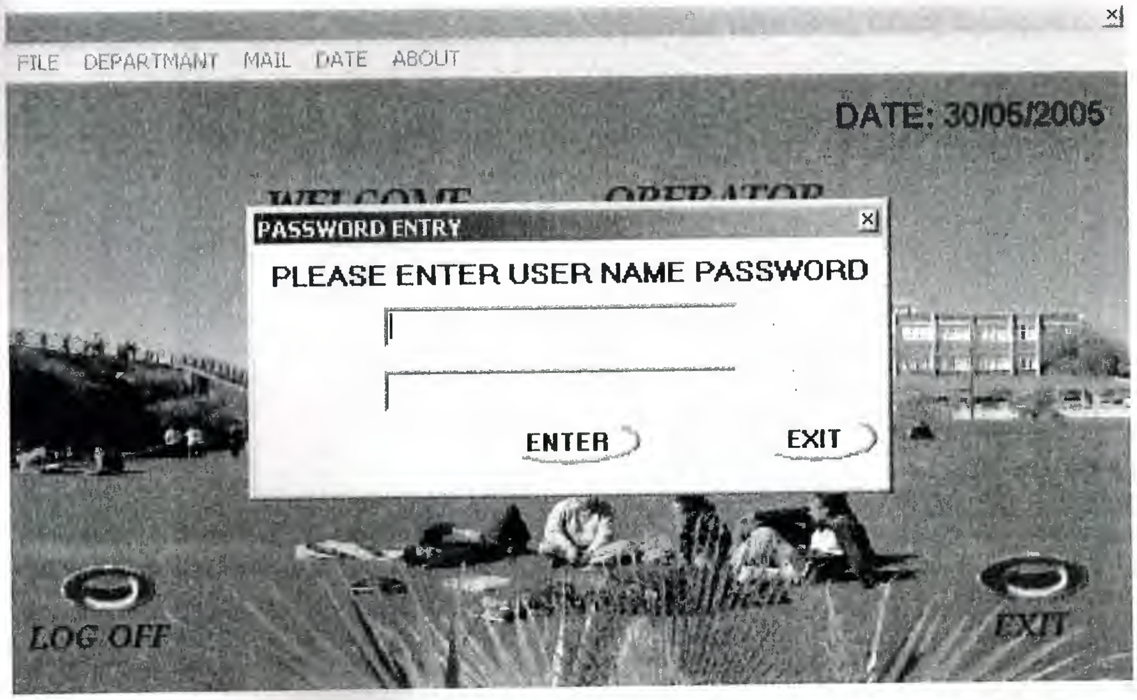
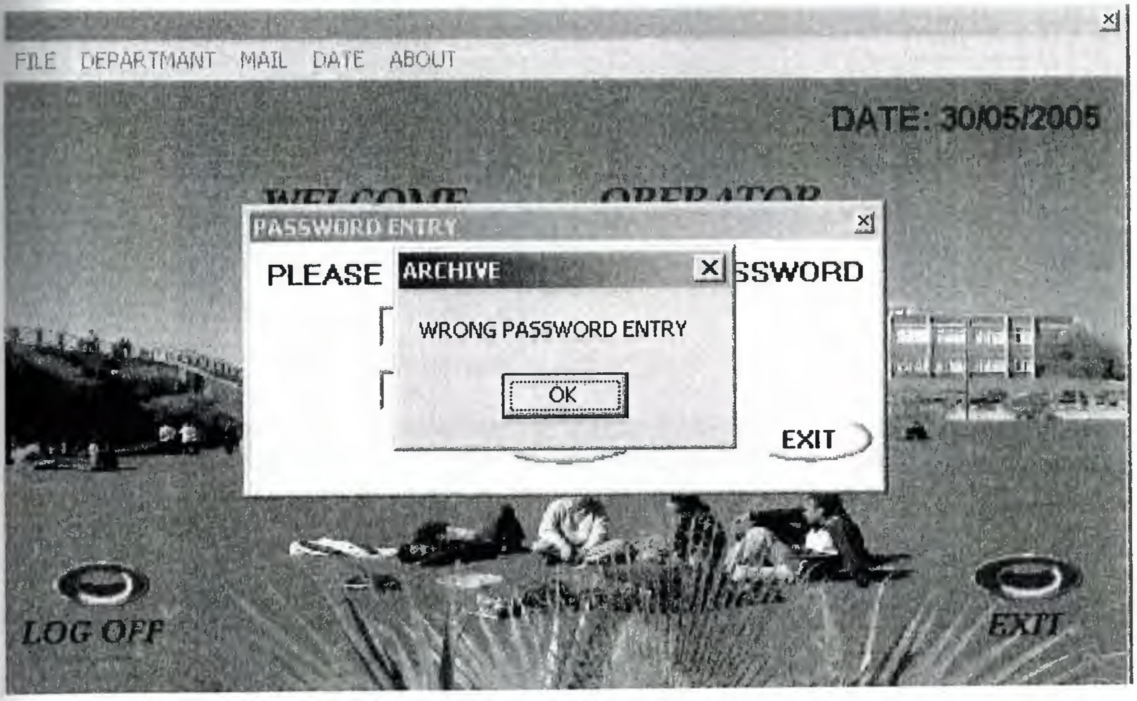


Fig 5.46

If you don't want username and password you can not enter this part shown in Fig 5.47 only the operator can enter this part. Operator is the headmaster of the program .Only operator can do what ever he/she want.



WRONG PASSWORD ENTRY

Fig 5.47

When you write correct username and password it will open user settings pages. In this part you give permissions for the users also you can add new user , delete the users and update the users. As you can see in the Fig 5.48.

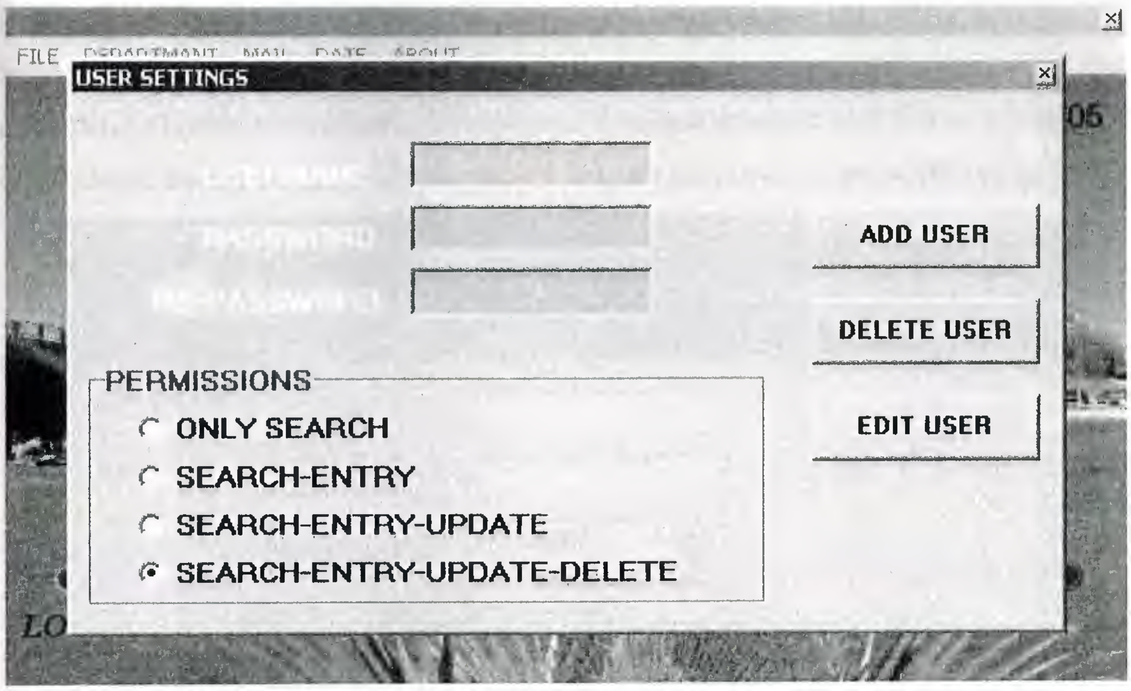


Fig 5.48

Personal entry from the department shown in Fig 5.49.when you click it will open personal entry's page shown in Fig 5.50

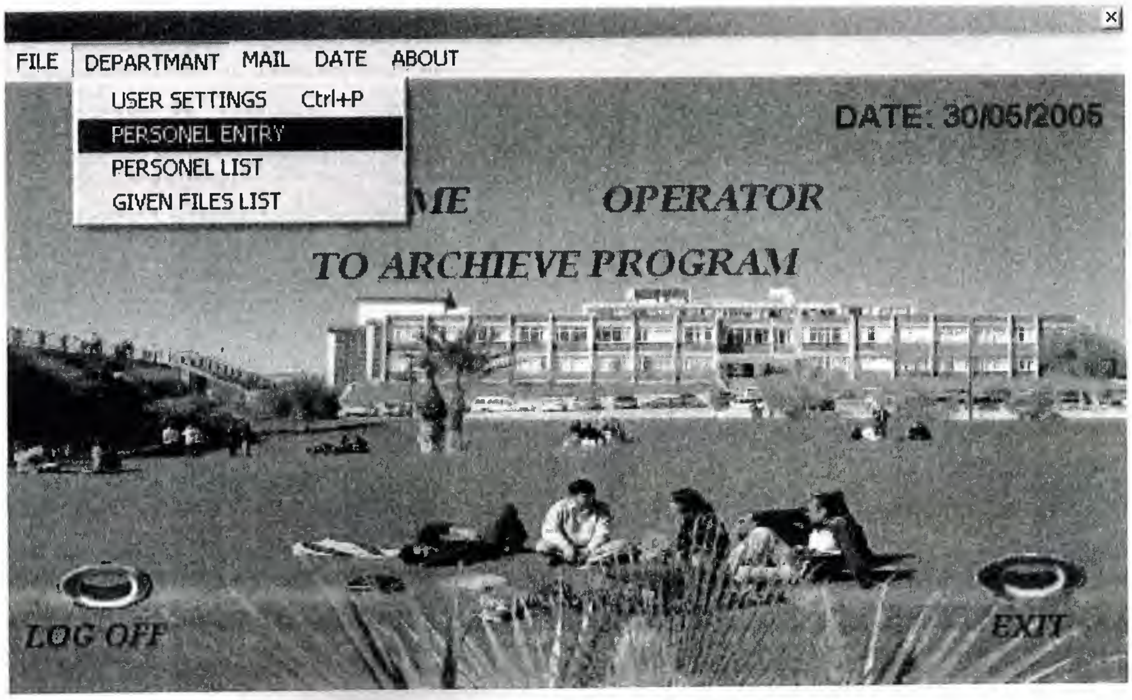
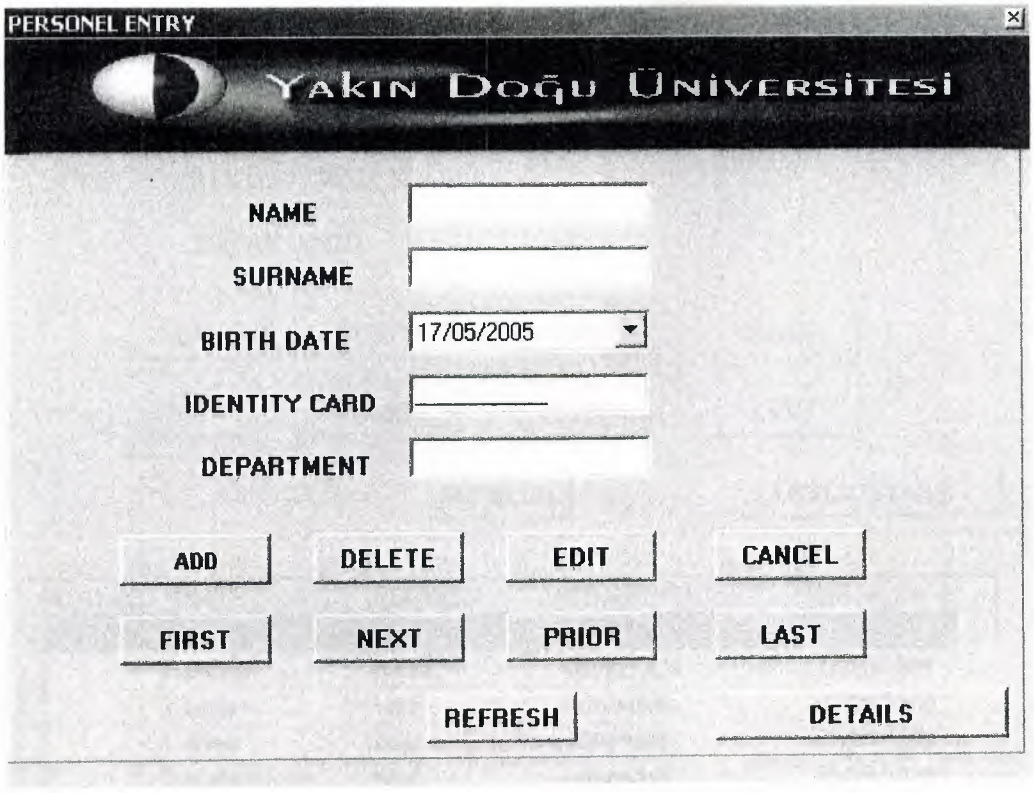


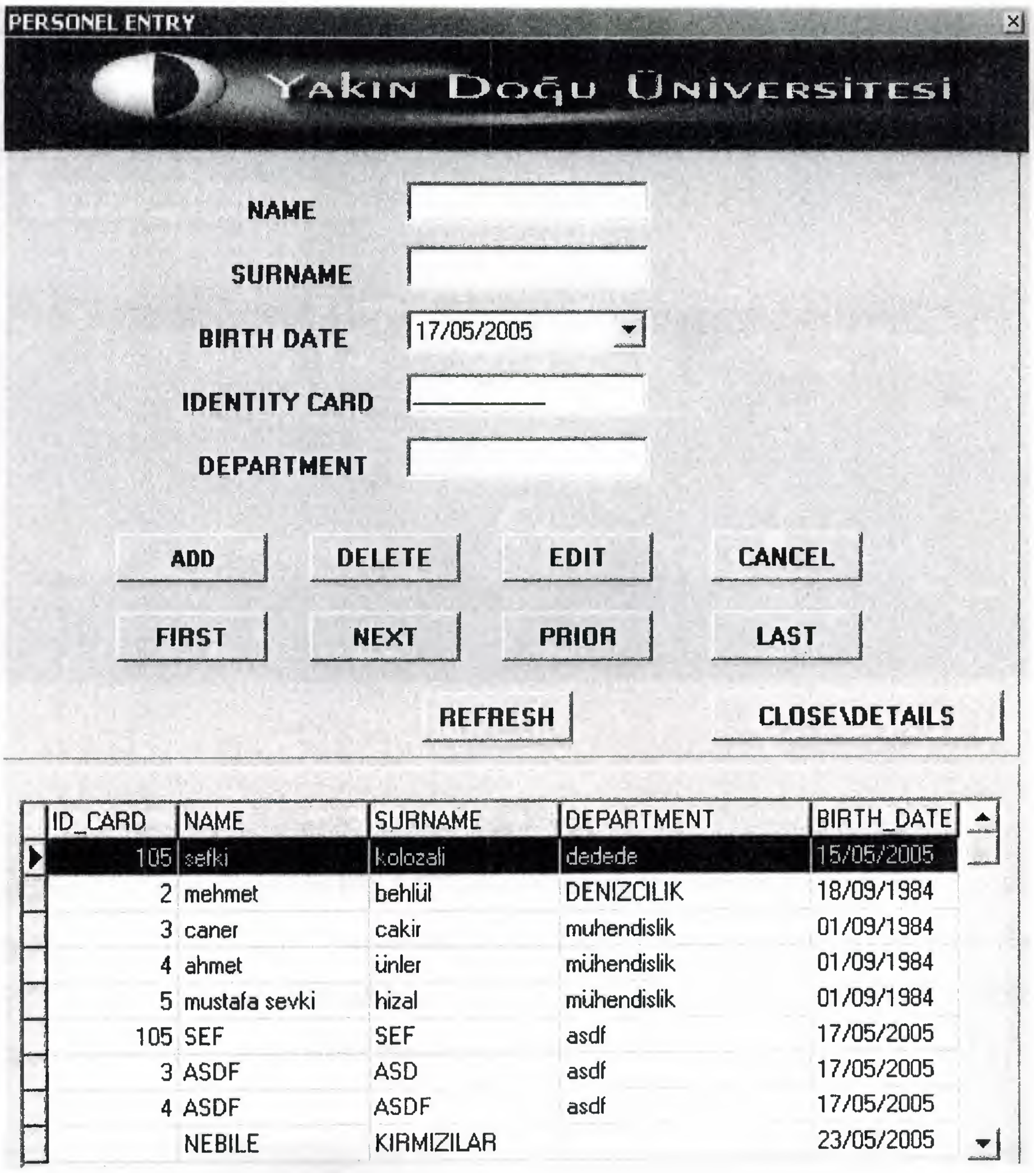
Fig 5.49

In the personal entry's part we can add ,delete and edit personals. For see the details we have to click details button in the personal entry's page. Its shown in Fig 5.50.when you click details button to see details it will opens details and button will change its name to close /detail for the close detail you will click again same button its shown in Fig 5.51



NAME	
SURNAME	
BIRTH DATE	17/05/2005
IDENTITY CARD	
DEPARTMENT	

Fig 5.50



BIRTH DATE: 17/05/2005

ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
105	sefki	kolozali	dedede	15/05/2005
2	mehmet	behlül	DENIZCILIK	18/09/1984
3	caner	cakir	muhendislik	01/09/1984
4	ahmet	ünler	mühendislik	01/09/1984
5	mustafa sevki	hizal	mühendislik	01/09/1984
105	SEF	SEF	asdf	17/05/2005
3	ASDF	ASD	asdf	17/05/2005
4	ASDF	ASDF	asdf	17/05/2005
	NEBILE	KIRMIZILAR		23/05/2005

Fig 5.51

The personal list part as you can see it Fig 5.52 when you click it, it will open new page its shown in Fig 5.53.

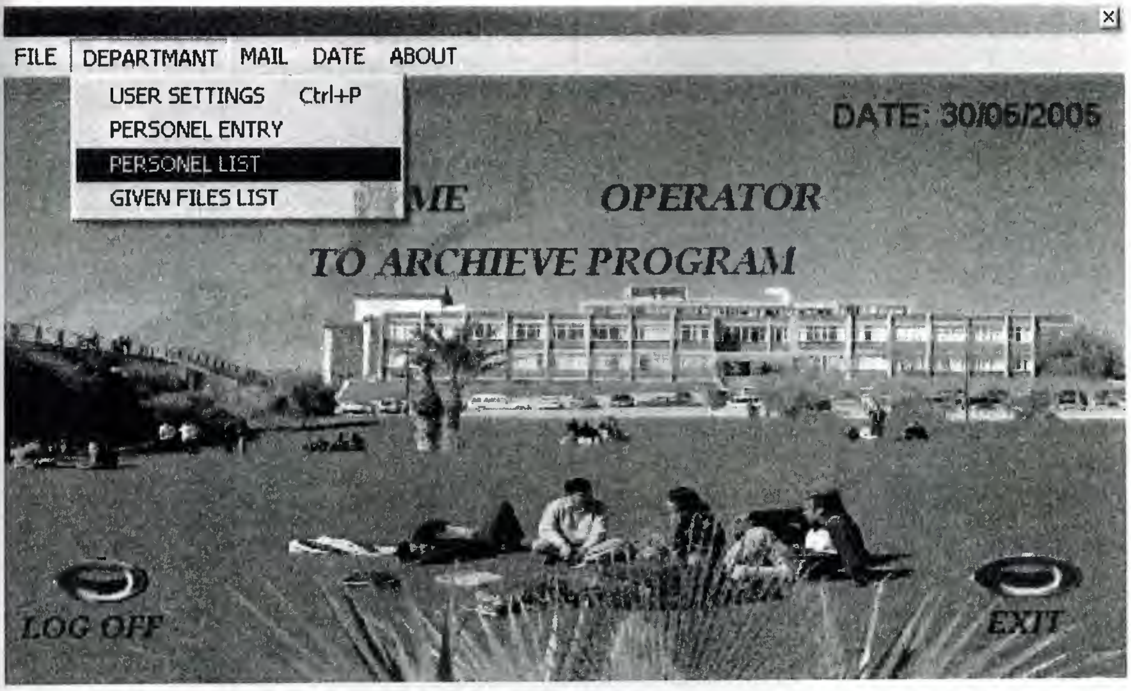


Fig 5.52

In personal list part shown in Fig 5.53 we can search the personal by id , name, surname , department and birth date .For see the list we have to click list option button as shown in Fig 5.54 and when we clicked it, it change its name to close/list and we can close the list from this button. If we want see all personal we have to click show all button.

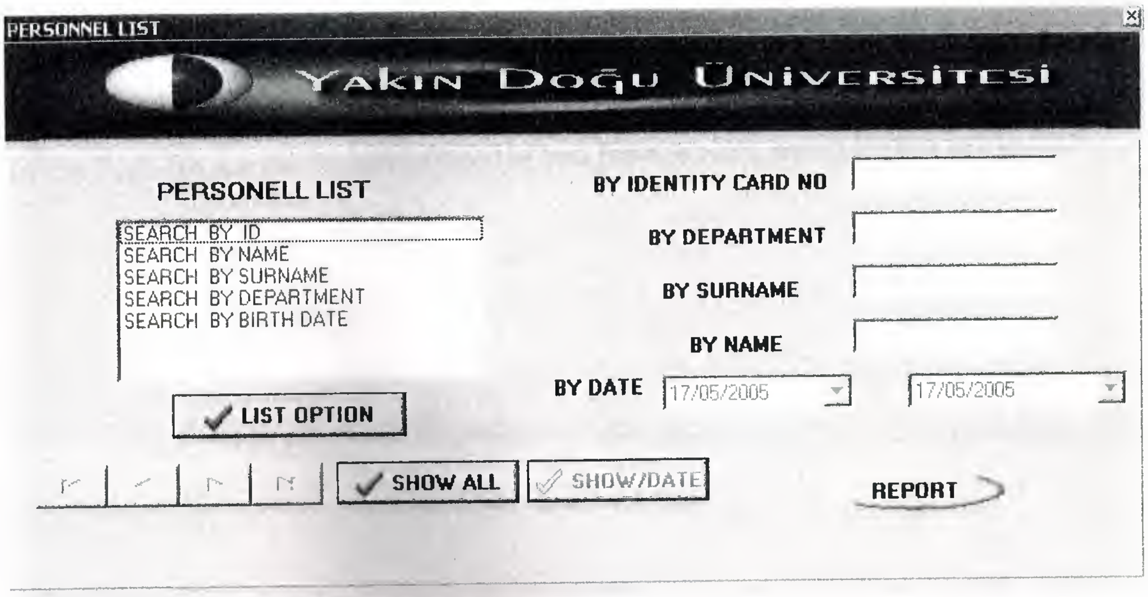
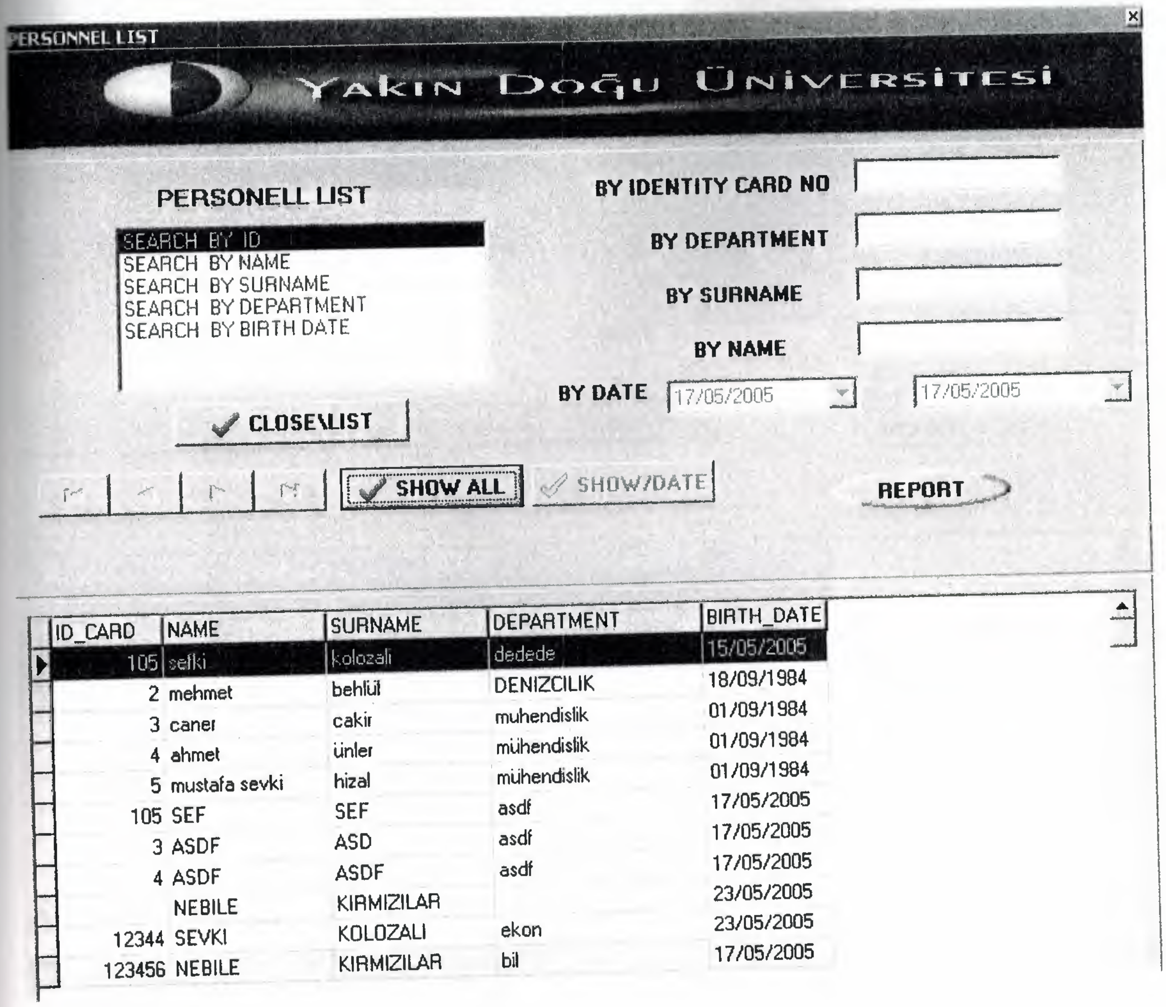


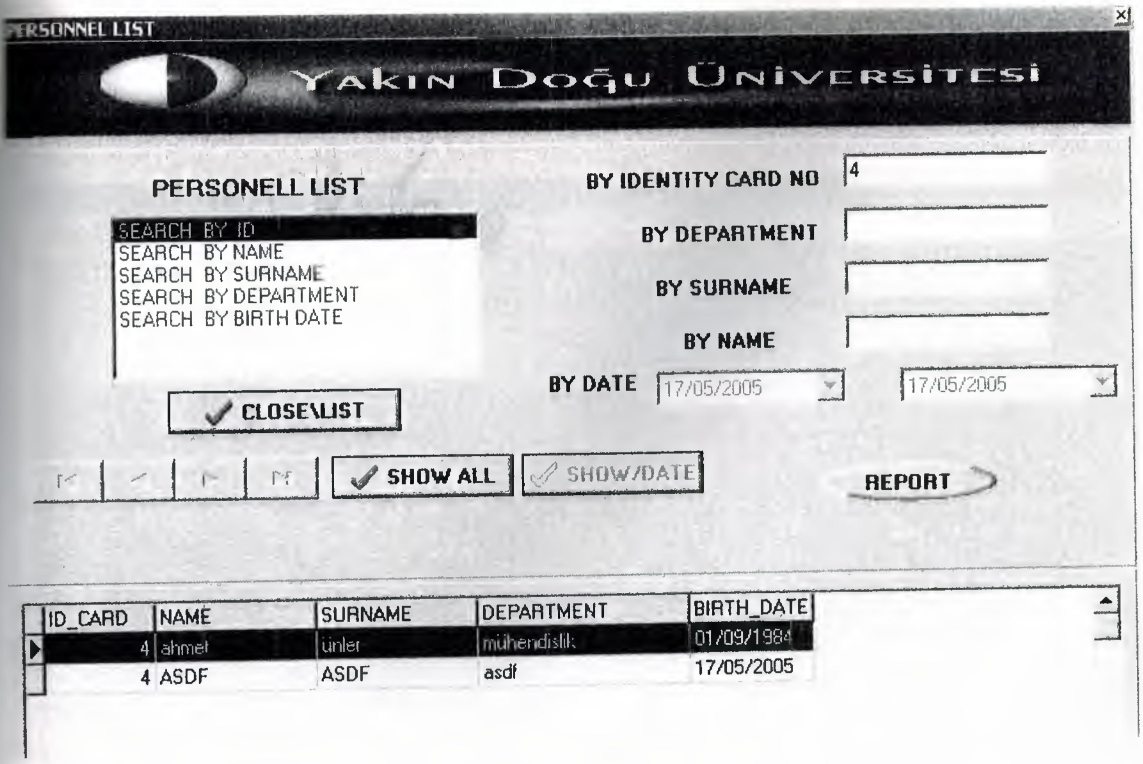
Fig 5.53



ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
105	sefki	kolozali	dedede	15/05/2005
2	mehmet	behlül	DENIZCILIK	18/09/1984
3	caner	cakir	muhendislik	01/09/1984
4	ahmet	ünler	mühendislik	01/09/1984
5	mustafa sevki	hizal	mühendislik	01/09/1984
105	SEF	SEF	asdf	17/05/2005
3	ASDF	ASD	asdf	17/05/2005
4	ASDF	ASDF	asdf	17/05/2005
	NEBILE	KIRMIZILAR		23/05/2005
12344	SEVKI	KOLOZALI	ekon	23/05/2005
123456	NEBILE	KIRMIZILAR	bil	17/05/2005

Fig 5.54

For make any search in this part we have to chose one part and make search its shown in Fig 5.55 .for see the reports of this list you have to click report button in Fig 5.55



PERSONELL LIST

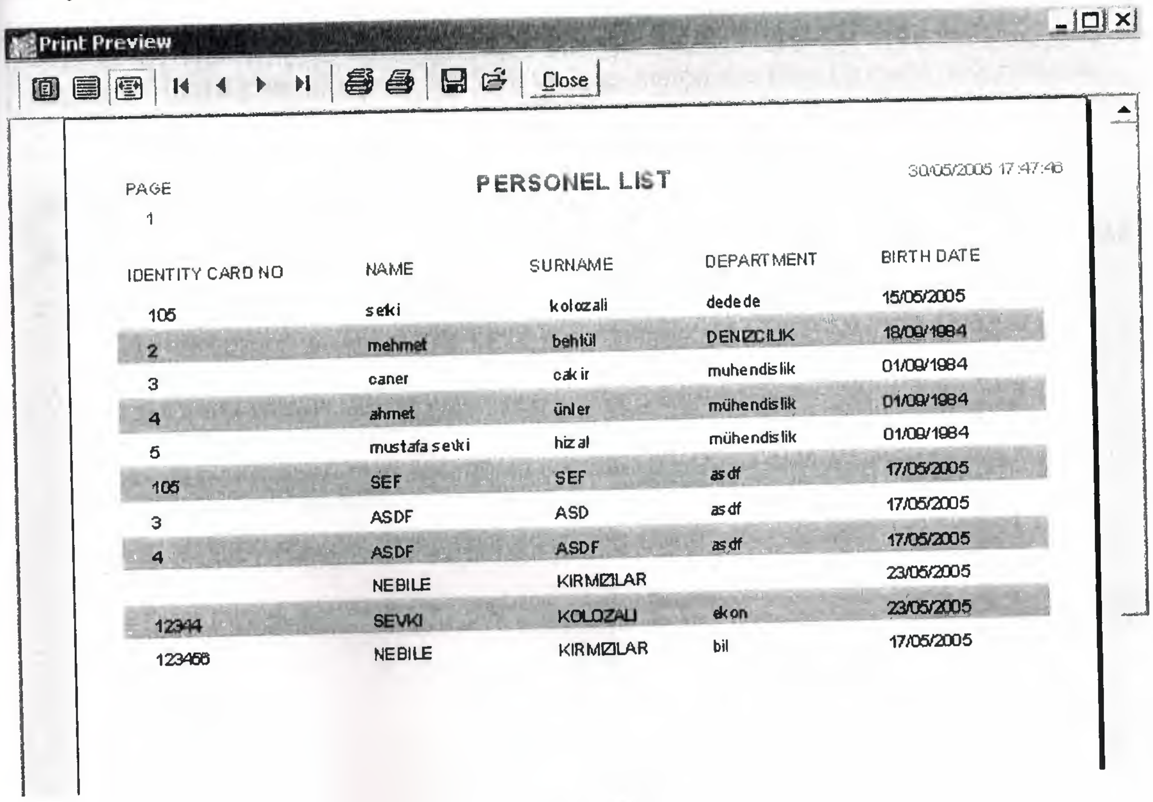
SEARCH BY ID
SEARCH BY NAME
SEARCH BY SURNAME
SEARCH BY DEPARTMENT
SEARCH BY BIRTH DATE

BY IDENTITY CARD NO: 4
BY DEPARTMENT:
BY SURNAME:
BY NAME:
BY DATE: 17/05/2005 17/05/2005

ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
4	ahmet	ünler	mühendislik	01/09/1984
4	ASDF	ASDF	asdf	17/05/2005

Fig 5.55

And you can see your reports as shown in Fig 5.56



PAGE
1

PERSONEL LIST

30/05/2005 17:47:46

IDENTITY CARD NO	NAME	SURNAME	DEPARTMENT	BIRTH DATE
105	seki	kolozali	dedede	15/05/2005
2	mehmet	behlül	DENIZCILIK	18/09/1984
3	caner	cakir	muhendislik	01/09/1984
4	ahmet	ünler	mühendislik	01/09/1984
5	mustafa sevki	hizal	mühendislik	01/09/1984
105	SEF	SEF	asdf	17/05/2005
3	ASDF	ASD	asdf	17/05/2005
4	ASDF	ASDF	asdf	17/05/2005
	NEBILE	KIRMIZILAR		23/05/2005
12344	SEVKI	KOLOZALI	ekon	23/05/2005
123456	NEBILE	KIRMIZILAR	bil	17/05/2005

Fig 5.56

Last part of the department menu is given file list its shown in Fig 5.57 when you click it.

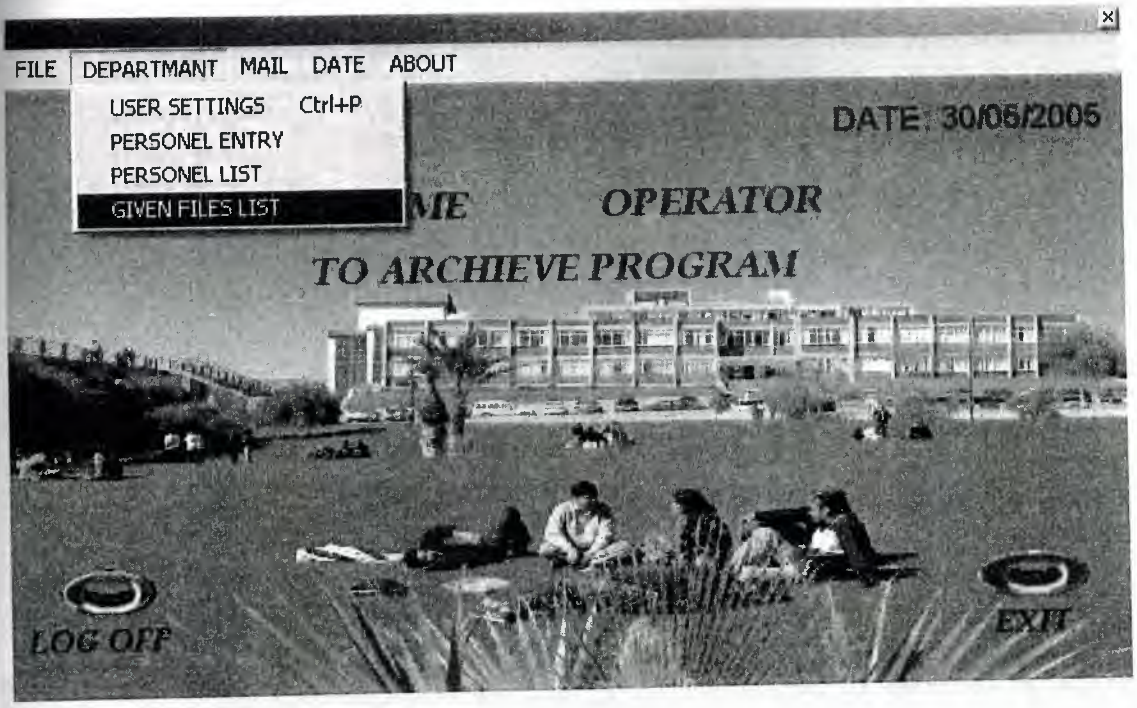


Fig 5.57

In given file list part shown in fig 5.58 we can search the files by code ,reference no , date , id , name and surname .For see the list we have to click list option button as shown in Fig 5.58 and when we clicked it ,it change its name to close/list and we can close the list from this button. If we want see all lists we have to click show all button. Shown in Fig 5.59.

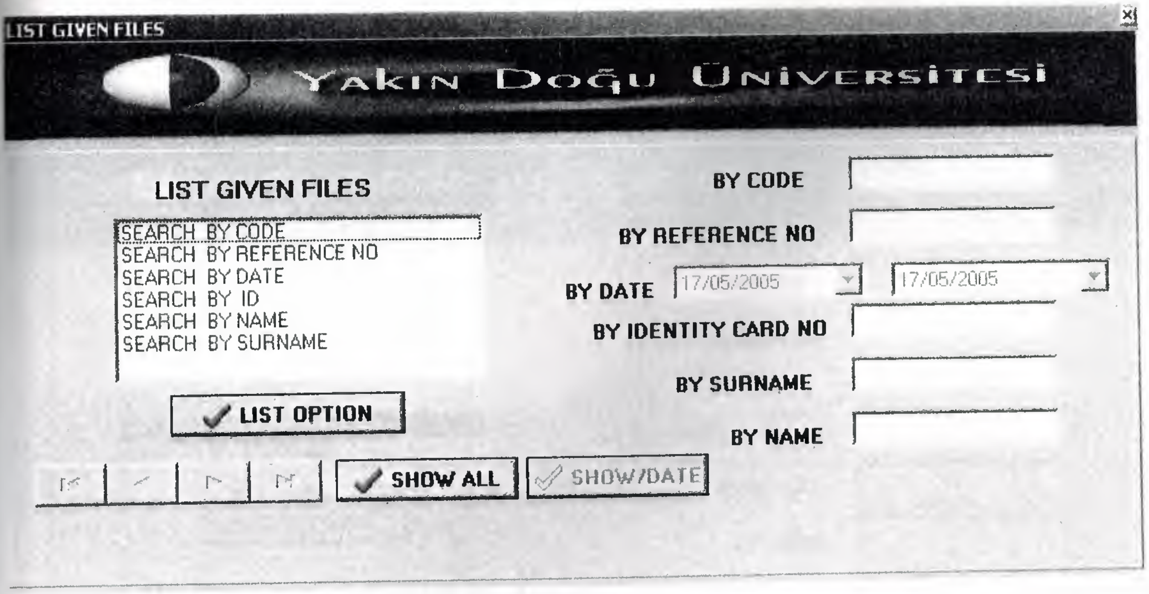
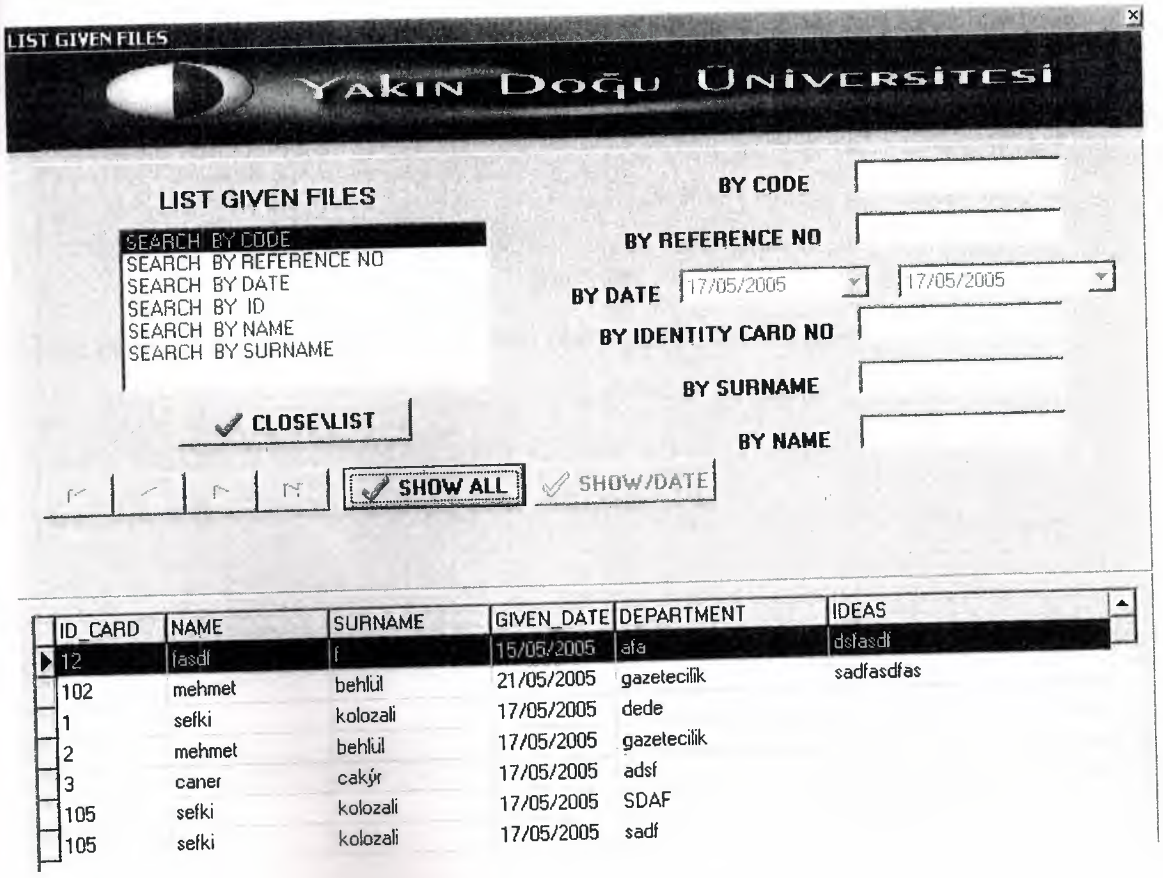


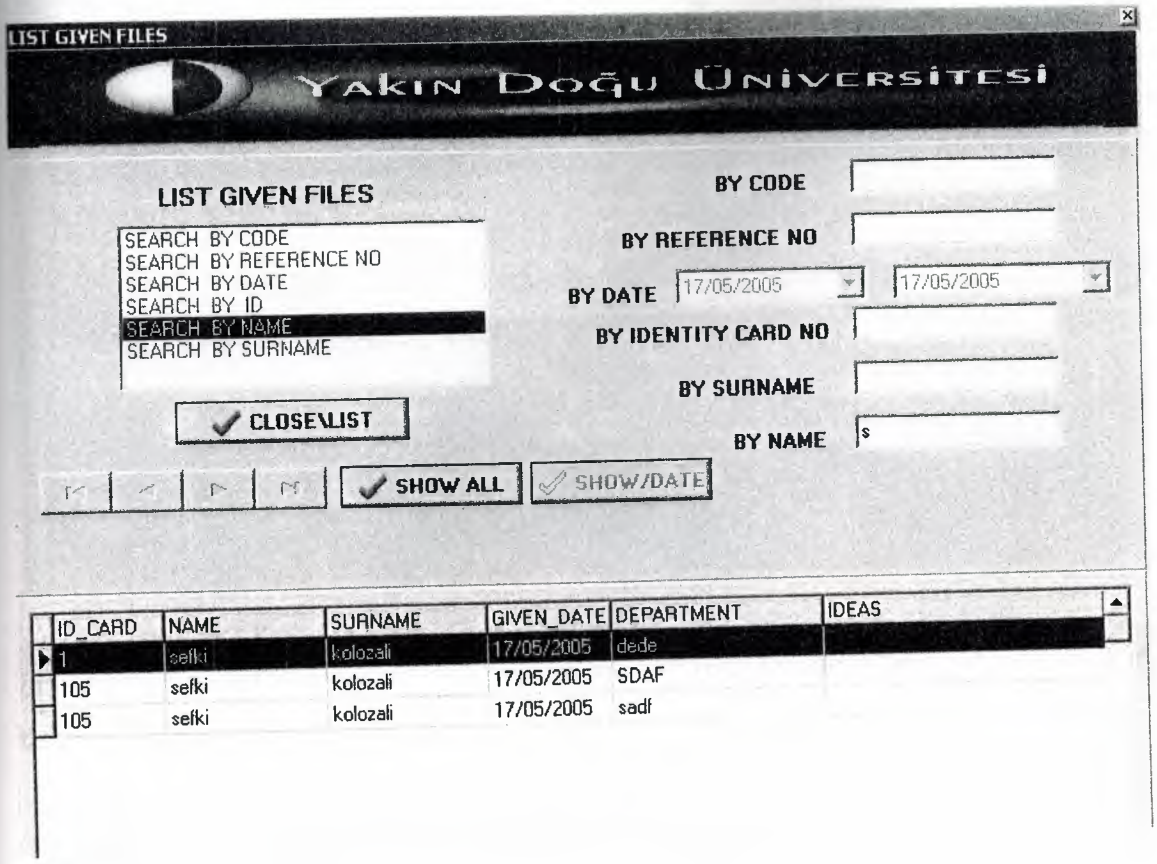
Fig 5.58



ID_CARD	NAME	SURNAME	GIVEN_DATE	DEPARTMENT	IDEAS
12	fasdf	f	15/05/2005	afa	dsfasdf
102	mehmet	behlül	21/05/2005	gazetecilik	sadfasdfas
1	sefki	kolozali	17/05/2005	dede	
2	mehmet	behlül	17/05/2005	gazetecilik	
3	caner	cakýr	17/05/2005	adsf	
105	sefki	kolozali	17/05/2005	SDAF	
105	sefki	kolozali	17/05/2005	sadf	

Fig 5.59

Which part as we like we will chose and write specification like shown in fig 5.60.



BY CODE

BY REFERENCE NO

BY DATE 17/05/2005 17/05/2005

BY IDENTITY CARD NO

BY SURNAME

BY NAME s

ID_CARD	NAME	SURNAME	GIVEN_DATE	DEPARTMENT	IDEAS
1	sefki	kolozali	17/05/2005	dede	
105	sefki	kolozali	17/05/2005	SDAF	
105	sefki	kolozali	17/05/2005	sadf	

Fig 5.60

Third part of the our program is the mail menu as shown in Fig 5.61.

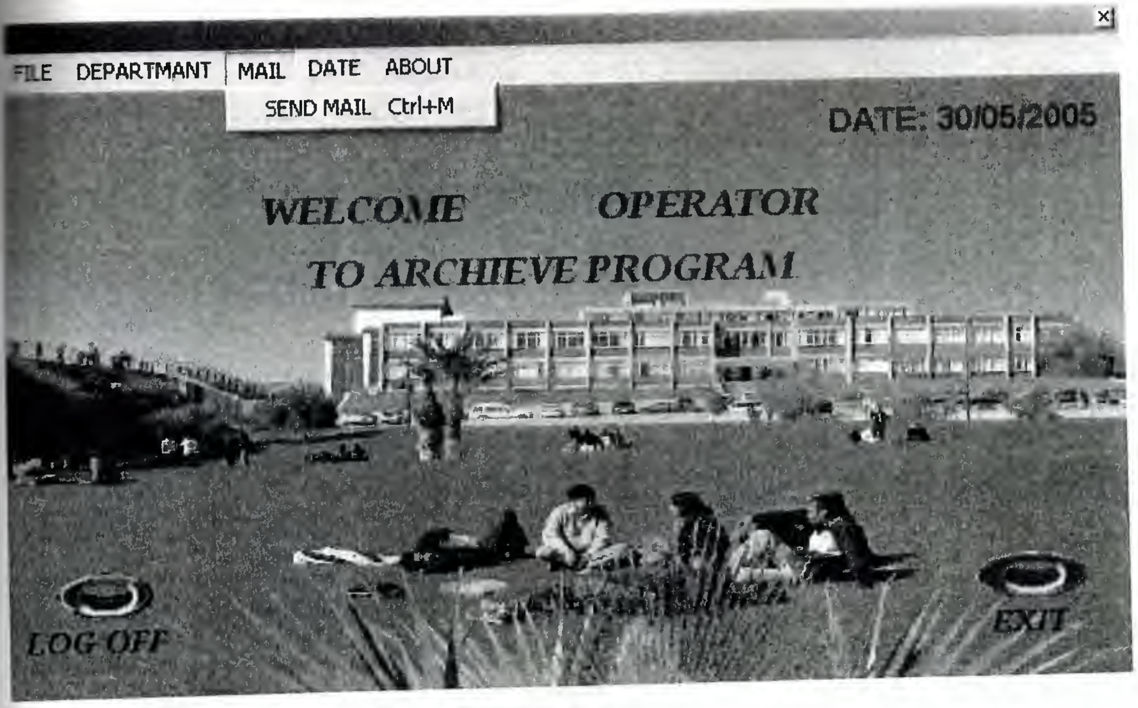


Fig 5.61

In our program without use outlook express we sent mail by using the send mail parts in the mail menus shown in Fig 5.62.

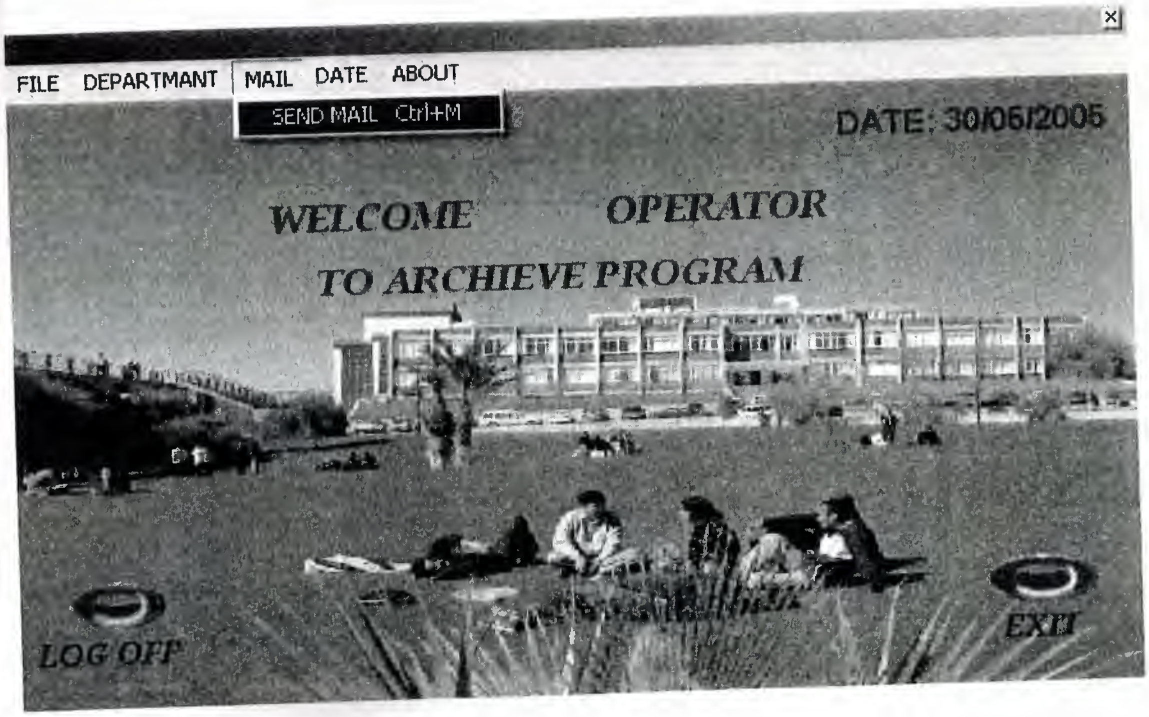
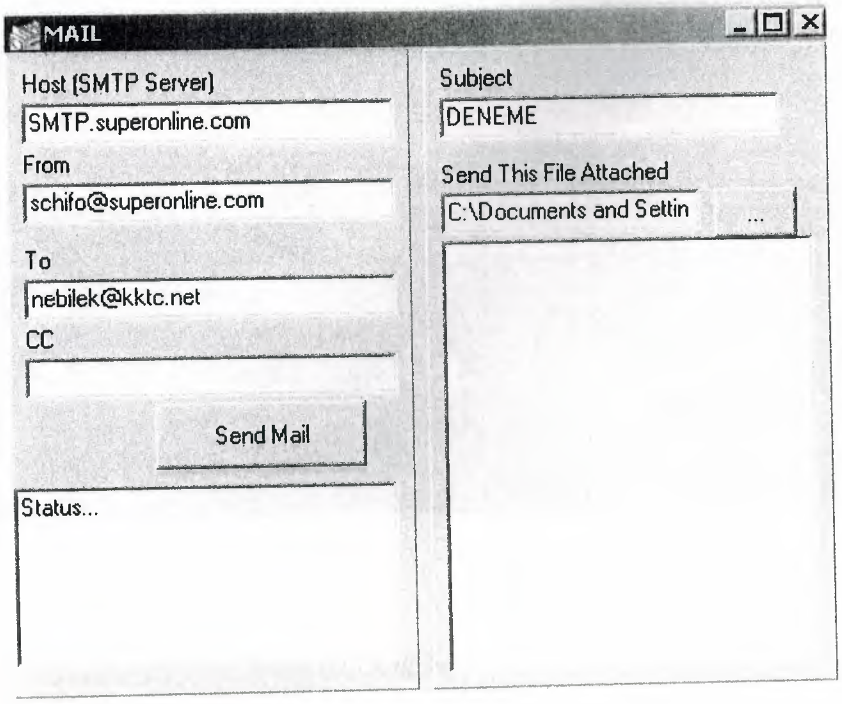


Fig 5.62

When we click the sent mail parts Fig 5.63 will open. With use this part we will send mail.



Host (SMTP Server): SMTP.superonline.com
From: schifo@superonline.com
To: nebilek@kktc.net
CC:
Subject: DENEME
Send This File Attached: C:\Documents and Settin

Fig 5.63

Firstly we have to write host its shown in Fig 5.63 it then click send mail button and send your mail.

The fourth menu in our program is the date menu's. In date's menu we have two part calendar and time parts as you see in Fig 5.64.

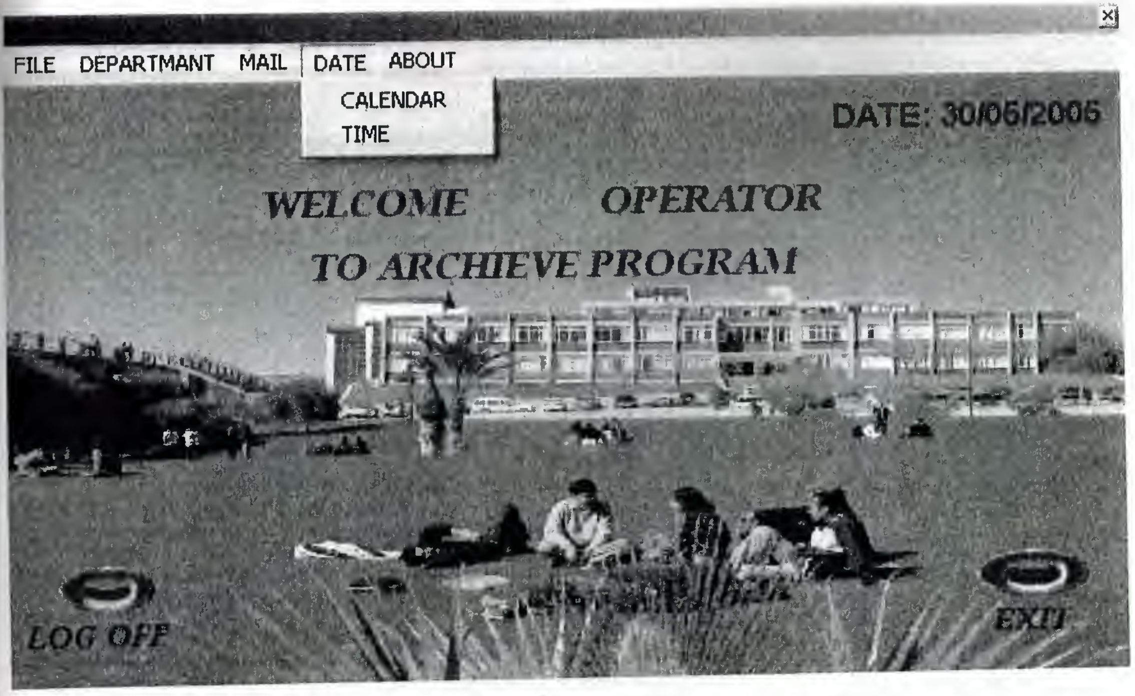


Fig 5.64

When you click the calendar has shown in fig 5.65 its open the calendar shown in Fig 5.66.

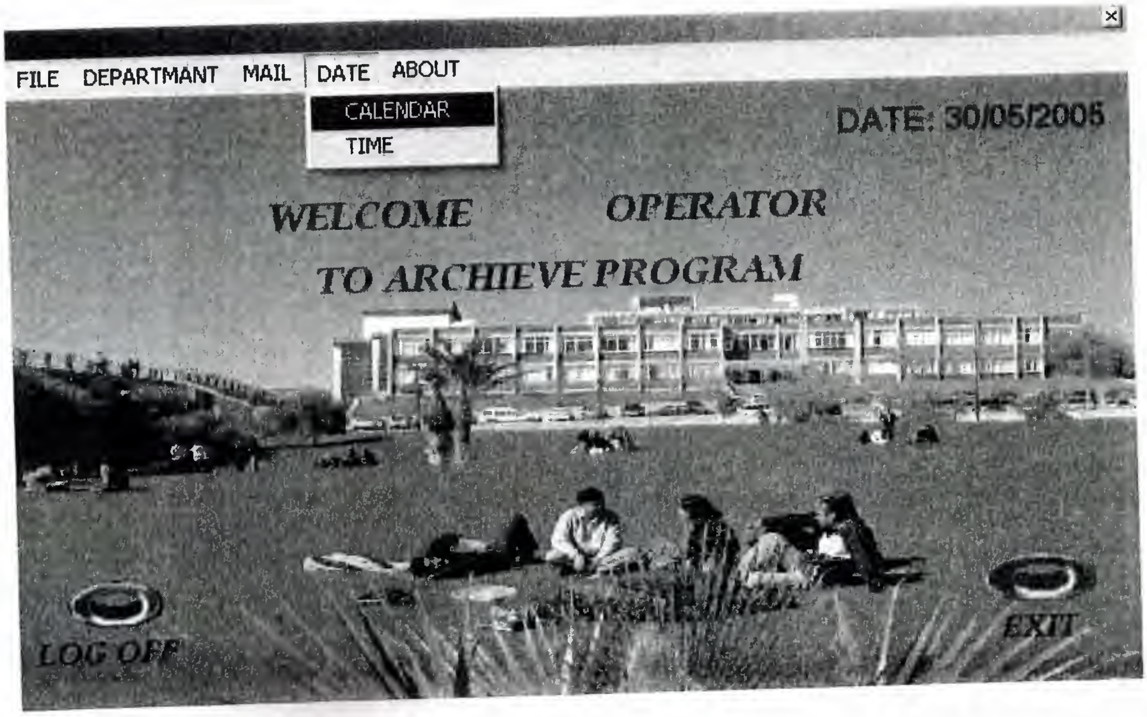
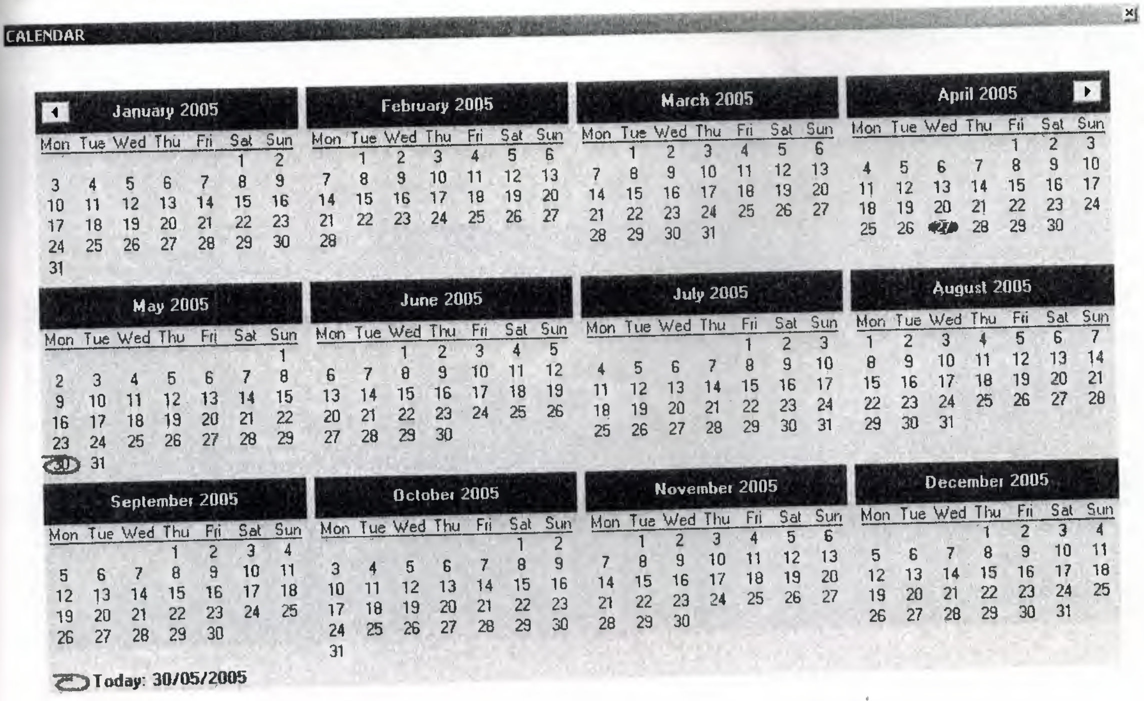


Fig 5.65


CALENDAR

January 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

February 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

March 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

April 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

May 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

June 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

July 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

August 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

September 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

October 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

November 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

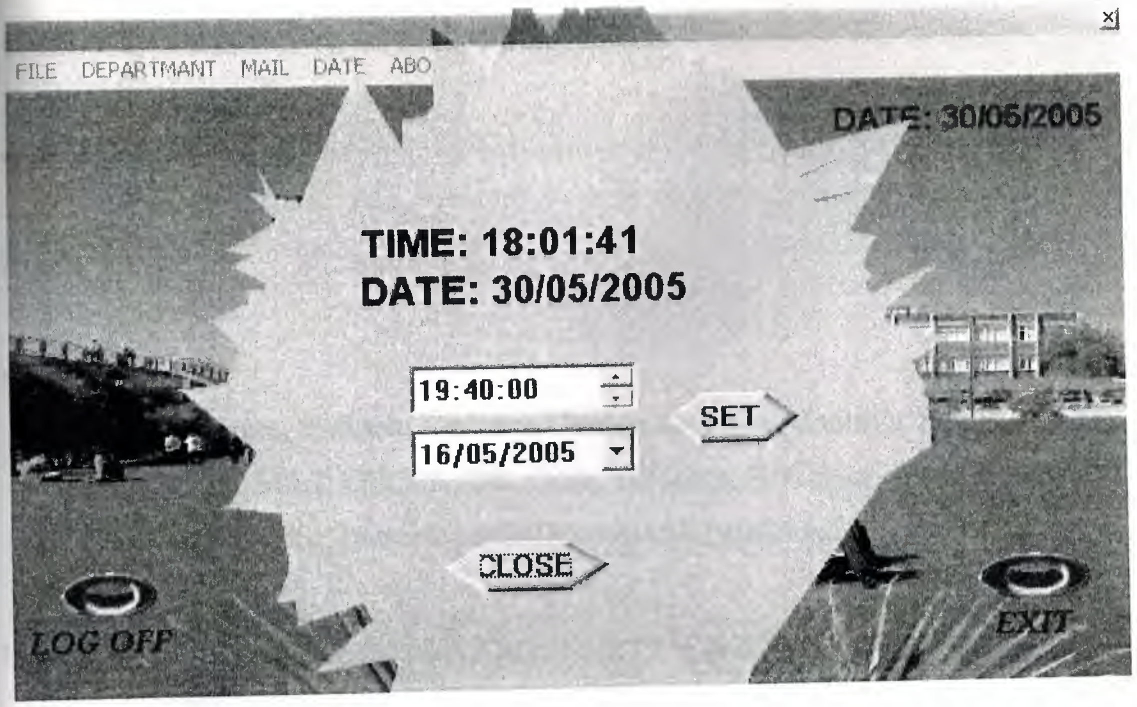
December 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Today: 30/05/2005

Fig 5.66

If we want to set system time we have to click time in date menus shown in Fig 5.67



TIME: 18:01:41
DATE: 30/05/2005

19:40:00
16/05/2005

Fig 5.67

The last menu is about.In the about menu we have some information about how program working and about us.

APPENDIX

```
unit NEW;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Mask, DBCtrls, DB, Grids, DBGrids, DBTables, ExtCtrls,
  ComCtrls,FileCtrl, jpeg, Buttons, janTriDownButton, janRoundButton;

type
  Tform2 = class(TForm)
    DataSource1: TDataSource;
    Panel1: TPanel;
    Panel2: TPanel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    BitBtn5: TBitBtn;
    BitBtn6: TBitBtn;
    BitBtn7: TBitBtn;
    BitBtn8: TBitBtn;
    BitBtn9: TBitBtn;
    Edit1: TEdit;
    Edit2: TEdit;
    DateTimePicker1: TDateTimePicker;
```

```
Edit4: TEdit;
BitBtn10: TBitBtn;
Edit5: TEdit;
DBGrid1: TDBGrid;
Panel3: TPanel;
Image1: TImage;
Panel4: TPanel;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
DBEdit5: TDBEdit;
DBEdit6: TDBEdit;
janRoundButton1: TjanRoundButton;
DBEdit7: TDBEdit;
janRoundButton2: TjanRoundButton;
procedure FormCreate(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure Edit4Change(Sender: TObject);
procedure Edit5Change(Sender: TObject);
procedure Edit1KeyPress(Sender: TObject; var Key: Char);
procedure Edit2KeyPress(Sender: TObject; var Key: Char);
procedure Edit4KeyPress(Sender: TObject; var Key: Char);
procedure Edit5KeyPress(Sender: TObject; var Key: Char);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn5Click(Sender: TObject);
procedure BitBtn6Click(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);
procedure BitBtn9Click(Sender: TObject);
```

```
    procedure BitBtn8Click(Sender: TObject);
    procedure BitBtn10Click(Sender: TObject);
    procedure MaskEdit1KeyPress(Sender: TObject; var Key: Char);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure BitBtn14Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn7Click(Sender: TObject);
    procedure janRoundButton1Click(Sender: TObject);
    procedure janRoundButton2Click(Sender: TObject);
    procedure DateTimePicker1KeyPress(Sender: TObject; var Key: Char);
    procedure Panel1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  form2: Tform2;

implementation

uses DataModule, CREATEFILE, GIRIS,EDIT, explorer, GIVE_FILE;

{$R *.dfm}

procedure Tform2.FormCreate(Sender: TObject);
begin
DateTimePicker1.date:=date;
form2.Width:=666;
form2.Height:=416;
```

```
edit1.Clear;
edit2.Clear;
edit4.Clear;
edit5.Clear;
end;

procedure Tform2.Button7Click(Sender: TObject);
begin
data.dosya1.last;
end;

procedure Tform2.Edit1Change(Sender: TObject);
begin
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit2Change(Sender: TObject);
begin
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit3Change(Sender: TObject);
begin
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit4Change(Sender: TObject);
begin
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit5Change(Sender: TObject);
begin
```

```
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
  if (SelStart = 0) or
    (Text[SelStart] = ' ') then
      if Key in ['a'..'z'] then
        Key := UpCase(Key);
    if key=#13 then edit2.setfocus;
if key=#13 then edit2.setfocus;
end;

procedure Tform2.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
  if (SelStart = 0) or
    (Text[SelStart] = ' ') then
      if Key in ['a'..'z'] then
        Key := UpCase(Key);
        if key=#13 then DateTimePicker1.setfocus;
if key=#13 then DateTimePicker1.setfocus;
end;

procedure Tform2.Edit4KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
  if (SelStart = 0) or
    (Text[SelStart] = ' ') then
      if Key in ['a'..'z'] then
        Key := UpCase(Key);
  if key=#13 then edit5.setfocus;
```

```
end;

procedure Tform2.Edit5KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
  if (SelStart = 0) or
    (Text[SelStart] = ' ') then
      if Key in ['a'..'z'] then
        Key := UpCase(Key);
  if key=#13 then BitBtn1.setfocus;
end;

procedure Tform2.BitBtn1Click(Sender: TObject);
var
btn:integer;
begin
if (edit1.Text='') or (edit2.Text='') or (edit4.Text='') or (edit5.Text='') then
 ShowMessage('PLEASE ENTER REPORT IDENTIFICATIONS')
 else
 begin
 DATA.dosya1.Filter:='CODE='+quotedstr(edit1.Text) + 'and
REF_NO='+quotedstr(edit2.Text);
 DATA.dosya1.Filtered:=true;

 if (data.dosya1.FieldValues['CODE']<>edit1.Text) and
(data.dosya1.FieldValues['REF_NO']<>edit2.Text) then
  begin
  btn:=MessageDlg('DO YOU WANT TO INSERT ANY FILE TO
PROGRAM?',mtWarning,[mbYes,mbNo,mbCancel],0);
   if btn=6 then
    begin
    forcedirectories('c:\DATAS\RECEIVED FILES\'+ edit1.text + '\' +edit2.text);
    form4.ShowModal;
```

NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ARCHIEVE PROGRAM

COM 400 PROJECT

ŞEFKİ KOLOZALİ (20010333)

NEBİLE KIRMIZILAR (20000325)

SUBMİTTED TO: ÜMİT SOYER

NICOSIA 2005

ACKNOWLEDGEMENT

We would like to thank to our supervisors Umit Soyer , without whom this project would have not been possible. Whose words of encouragement kept us doing our project. His invaluable knowledge for the project has made our taking keen interest in our project.

Specially we would like to thank Hasan Yucel who helped us so much in doing our project. We would also like to thank Caner Cakir for his helped, he was always near us and helped us. Also thank for all our friend to be belive us and give as moral: Yazen Al Shawneh , Sule Akkurt ,Mustafa Sevki Hizal ,Duygu Ozsen and Ahmet Unler. Being with them make our time full of fun. They encouraged us allot in completing our project. We want to thank them as they contributed their time and provided very helpful suggestions.

Finally, we want to thank our family, especially our parents. Without their endless support and love for us, we would never achieve our current position.

ABSTRACT

The archive system is the principles of organization and description. It use for to find our documents more easy than looking step by step all the documents. In our project we did an archive program to fine documents and images more easy without looking anywhere just write name of what you are looking and search so it will be more easy to find what we are looking. With out move your place you can find what ever you are looking. But all of this making with controlling. But we have to know how we will fine. Research is work that tries to discover facts about something. if there is resemblance between two things there similar to each other. if you research something ,you try to discover facts about it.

TABLE OF CONTENTS

ACKNOWLEDGEMENT.. i
ABSTRACT..ii
TABLE OF CONTENTS..iii
INTRODUCTION..viii

CHAPTER 1 1

THE ARCHITECTURE OF DATABASE APPLICATIONS.. 1

1.1 ARCHITECTURE OF DATABASE.. 1

1.1.1 RELATIONAL DATABASE CONCEPTS.. 1

1.1.1.1 DATABASE TABLES 1

1.1.1.1.1 CHARACTERISTICS OF RELATIONAL TABLES........................2

1.1.1.1.2 WHAT DO YOU KEEP IN EACH TABLE? 3

1.1.1.2 RELATIONS BETWEEN TABLES 3

1.1.1.2.1 TABLES HAVE A PRIMARY KEY 4

1.1.1.2.1.1 EXAMPLES..4

1.1.1.2.2 TABLES ARE RELATED BY FOREIGN KEYS 4

1.1.1.2.2.1 EXAMPLE..5

1.1.1.3 OTHER DATABASE OBJECTS 5

1.1.2 THE PIECES OF A DATABASE SYSTEM 7

1.1.3 HOW THE PIECES FIT TOGETHER..9

1.1.3.1 PERSONAL APPLICATIONS AND EMBEDDED DATABASES 9

1.1.3.2 CLIENT/SERVER APPL. AND MULTI-USER DATABA.................10

1.1.4 MULTI-TIER COMPUTING ARCHITECTURE..11

1.1.5 USING MULTIPLE DATABASES 12

1.1.5.1 RUNNING DATABASES ON A SINGLE DATABASE SERVER...12

1.1.5.2 ACCESSING DATA IN OTHER DATABASES 13

1.2 ABOUT DBASE AND PARADOX 13

1.2.1 DBASE IV TABLE SPECIFICATION 13

1.2.2 DBASE V TABLE SPECIFICATIONS 14

1.2.3 DBASE FIELD TYPES 14

1.2.4 PARADOX STANDARD TABLE SPECIFICATIONS 15

1.2.5 PARADOX 5 TABLE SPECIFICATIONS 16

1.2.6 PARADOX 7 AND ABOVE TABLE SPECIFICATIONS 17

1.2.7 PARADOX FIELD TYPES 17

1.2.8 ABOUT OUR DATABASE **19**

CHAPTER 2 22

OPERATION SYSTEM 22

2.1 OPERATING SYSTEM 22

2.2 OPERATING SYSTEM TYPES 22

2.2.1 MULTI-USER 23

2.2.2 MULTIPROCESSING 23

2.2.3 MULTITASKING 23

2.2.4 MULTITHREADING 23

2.3 INTRODUCTION TO UNIX: 24

2.3.1 OBJECTIVES 24

2.3.2 WHAT IS AN OPERATING SYSTEM? 24

2.3.3 A BRIEF HISTORY OF UNIX 26

2.3.4 ARCHITECTURE OF THE LINUX OPERATING SYSTEM 29

2.4 AN INTRODUCTION TO LINUX 31

2.4.1 WHAT IS LINUX? 31

2.4.2 WHY USE LINUX? 32

2.4.2.1 CONFIGURABILITY 32

2.4.2.2 CONVENIENCE 33

2.4.2.3 Stability 33
2.4.2.4 Community 33
2.4.2.5 Freedom 34
2.5 The differences between Linux and other UNIX 34
CHAPTER 3 .... ***35***
DELPHI........ *35*
3.1.INTRODUCTION........ *35*
3.2. ASYNCHRONOUS INTERACTION *37*
3.3 ANONYMITY *40*
3.4 MODERATION AND FACILITATION *43*
3.5 STRUCTURE *44*
3.5.1 The Policy Delphi *45*
3.5.2 The Trend Model *47*
3.6 ANALYSIS........ *50*
3.6.1 Scaling Methods *50*
3.6.2 Structural Modeling........ *53*
3.7 DELPHI, EXPERT SYS., GDSS, AND COLLABORATIVE SYSTEMS..... 55
3.7.1 The Deductive Level of Disagreements *57*
3.7.2 The Inductive Level of Disagreements *58*
3.7.3 Goal and Value Disagreements *59*
3.8 CONCLUSION *60*
CHAPTER 4 .... **61**
ARCHIVE SYSTEM 61
4.1 Basic concepts and principles of archives and records management 62
4.1.1 The Inter-relatedness of records 62
4.1.2 The central importance of context........ 62
4.1.3 The function of records as evidence 63
4.1.3.1 Recordness requires that records are: 63
4.1.3.1.1 Complete........ 64

4.1.3.1.2 'Fixed' 64

4.1.3.1.3 Organic 64

4.1.3.1.4 Contextual 64

4.1.3.1.5 Authoritative/'Official' 65

4.1.3.1.6 Unique 65

4.1.3.1.6.1. The principle of provenance 65

4.1.3.1.6.2. The principle of original order 65

4.1.3.1.6.3. The chain of responsible custody 67

4.1.3.1.6.3.1 What goes on in the archives? 67

4.1.3.1.6.3.2 CADSS 67

4.1.3.1.6.3.3 CONTROL 68

4.1.3.1.6.3.4 ACCESSIBILITY 69

4.1.3.1.7 DISPOSAL 71

4.1.3.1.8 STORAGE 72

4.1.3.1.9 Check Answer 73

4.1.3.1.10 SUSTAIN 75

4.1.3.1.11 Recordkeeping regimes 77

4.1.3.1.12 Where we work: The recordkeeping context 77

4.1.3.1.12.1 Public sector placements 77

4.1.3.1.12.2 Private sector placements 78

4.1.3.1.13 Type and functional emphasis 78

4.1.3.1.14 Accountable recordkeeping regimes at macro level. 79

4.1.3.1.15 Recordkeeping rewards 82

4.1.3.1.16 Recordkeeping and related professions 82

4.1.3.1.16.1 What's the difference? 82

4.1.3.1.17 Recordkeeping and specialist recordkeepers 83

CHAPTER 5 .... **84**

ARCHIVE PROGRAM 84

CONCLUSION 157

REFERNCES……………………………………………………………………157

INTRODUCTION

Archive system is main base registration function of office management for extension. For correct registration of incoming and outgoing files or documents you need always true programming system. Our program is using computer for registration sending and receiving documents from selected office. Documents are enter to the Archive system with their attachments. Any one can easily reach all of enclosed file or documents.

Our first chapter is all about explaining Architecture of database systems. About relational database's concepts, the pieces of database systems, how the pieces fit together, multi-tier computing architecture and multiple databases, dbase and paradox.

Our second chapter is all about giving details of operation systems. Which are Unix, Linux and the differential of Unix and Linux.

Our third chapter is all about Delphi. Introduction of Delphi, asynchronous interaction ,anonymity and so on.

Our fourth chapter we explained about the archive systems. Basic concepts and principles of archives and how its work.

Our last fifth chapter is about the our program archive program how its wo.how we can use it and so on.

CHAPTER 1

The Architecture of Database Applications

1.1 Architecture of database

1. Relational database concepts
2. The pieces of a database system
3. How the pieces fit together
4. Multi-tier computing architecture
5. Using multiple databases
6. About dbase

1.1.1 Relational database concepts

A relational database-management system (RDBMS) is a system for storing and retrieving data, in which the data is organized into interrelated tables.

SQL Anywhere Studio provides two relational database systems. Adaptive Server Anywhere is the primary, full featured RDBMS, with a multitude of uses, from a network database server hosting many clients to a compact embedded database. UltraLite is a small-footprint relational database. The UltraLite deployment technology allows you to use Adaptive Server Anywhere features on even the smallest of devices.

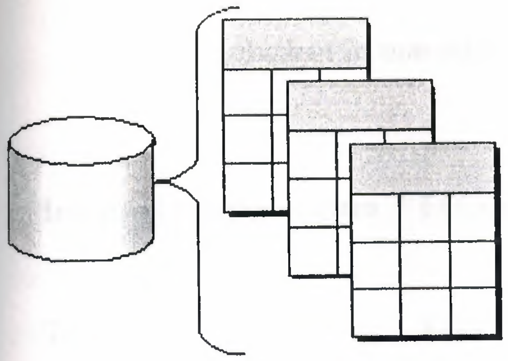
1.1.1.1 Database tables

1.1.1.2 Relations between tables

1.1.1.3 Other database objects

1.1.1.1 Database tables

In a relational database, all data is held in tables, which are made up of rows and columns.



Each table has one or more columns, and each column is assigned a specific data type, such as an integer, a sequence of characters (for text), or a date. Each row in the table has a single value for each column.

For example, a table containing employee information may look as follows:

emp_ID	**emp_lname**	**emp_fname**	**emp_phone**
10057	Huong	Zhang	1096
10693	Donaldson	Anne	7821

1.1.1.1.1 Characteristics of relational tables

The tables of a relational database have some important characteristics:

- There is no significance to the order of the columns or rows.
- Each row contains one and only one value for each column, or contains NULL, which indicates that there is no value for that column.
- All values for a given column have the same data type.

The following table lists some of the formal and informal relational database terms describing tables and their contents, together with their

equivalent in non-relational databases. This manual uses the informal terms.

Informal relational term	**Formal relational term**	**Non-relational term**
Table	Relation	File
Column	Attribute	Field
Row	Tuple	Record

1.1.1.1.2 What do you keep in each table?

Each table in the database should hold information about a specific thing, such as employees, products, or customers.

By designing a database this way, you can set up a structure that eliminates redundancy and the possible inconsistencies caused by redundancy. For example, both the sales and accounts payable departments might enter and look up information about customers. In a relational database, the information about customers is stored only once, in a table that both departments can access.

1.1.1.2 Relations between tables

You use primary keys and foreign keys to describe relationships between the information in different tables. Primary keys identify each row in a table uniquely, and foreign keys define the relationships between rows in different tables.

Primary keys and foreign keys let you use relational databases to hold information in an efficient manner, without redundancy.

1.1.1.2.1 Tables have a primary key

1.1.1.2.2 Tables are related by foreign keys

1.1.1.2.1 Tables have a primary key

Each table in a relational database should have a primary key. The primary key is a column, or set of columns, that uniquely identifies each row. No two rows in a table may have the same primary key value

.

1.1.1.2.1.1 Examples

In the sample database, the employee table stores information about employees. It has a primary key column named emp_id, which holds a unique ID number assigned to each employee. A single column holding an ID number is a common way to assign primary keys, and has advantages over names and other identifiers that may not always be unique.

A more complex primary key can be seen in the sales_order_items table of the sample database. The table holds information about individual items on orders from the company, and has the following columns:

- id An order number, identifying the order the item is part of.
- line_id A line number, identifying each item on any order.
- prod_id A product ID, identifying the product being ordered.
- quantity A quantity, displaying how many items were ordered.
- ship_date A ship date, displaying when the order was shipped.

A particular sales order item is identified by the order it is part of, and by a line number on the order. These two numbers are stored in the id and line_id columns. Items may share a single id value (corresponding to an order for more than one item) or they may share a line_id number (all first items on different orders have a line_id of 1). No two items share both values, and so the primary key is made up of these two columns.

1.1.1.2.2 Tables are related by foreign keys

The information in one table is related to that in other tables by foreign keys.

1.1.1.2.2.1 Example

The sample database has one table holding employee information and one table holding department information. The department table has the following columns:

- dept_id An ID number for the department. This is the primary key for the table.
- dept_name The name of the department.
- dept_head_id The employee ID for the department manager.

To find the name of a particular employee's department, there is no need to put the name of the employee's department into the employee table. Instead, the employee table contains a column holding a number that matches one of the dept_id values in the department column.

The dept_id column in the employee table is called a foreign key to the department table. A foreign key references a particular row in the table containing the corresponding primary key.

In this example, the employee table (which contains the foreign key in the relationship) is called the foreign table or referencing table. The department table (which contains the referenced primary key) is called the primary table or the referenced table.

1.1.1.3 Other database objects

There is more to a relational database than simply a set of related tables. You will also find the following objects in a relational database:

- Indexes Indexes allow quick lookup of information. Conceptually, an index in a database is like an index in a book. In a book, the index relates each indexed term to the page or pages on which that word appears. In a database, the index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored.

Indexes are an important design element for high performance. You must usually create indexes explicitly, but indexes for primary and foreign keys and for unique columns are created automatically. Once created, the use of indexes is transparent to the user.

- Views   Views are computed tables, or virtual tables. They look like tables to client applications, but they do not hold data. Instead, whenever they are accessed, the information in them is computed from the underlying tables.

  The tables that actually hold the information are sometimes called base tables to distinguish them from views. A view is defined with a SQL query on base tables or other views.

- Stored procedures and triggers   These are routines held in the database itself that act on the information in the database.

  You can create and name your own stored procedures to execute specific database queries and to perform other database tasks. Stored procedures can take parameters. For example, you might create a stored procedure that returns the names of all customers who have spent more than the amount that you specify as a parameter in the call to the procedure.

  A trigger is a special stored procedure that automatically fires whenever a user updates, deletes, or inserts data, depending on how you define the trigger. You associate a trigger with a table or columns within a table. Triggers are useful for automatically maintaining business rules in a database.

- Users and groups   Each user of a database has a user ID and password. You can set permissions for each user so that confidential information is kept private and users are prevented from making unauthorized changes. Users can be assigned to groups in order to make the administration of permissions easier.

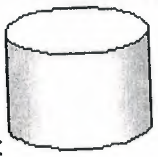
- Java objects   You can install Java classes into the database. Java classes provide a powerful way of building logic into your

database, and a special class of user-defined data types for holding information.

All of these items together make up a relational database-management system (RDBMS); a system for storing and retrieving data, in which the data is organized into interrelated tables.

1.1.2 The pieces of a database system

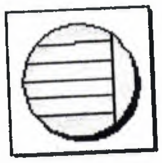
Relational database management systems contain the following pieces:

A database Data is stored in a database. In diagrams in the documentation, a database is indicated by a cylinder: 

An Adaptive Server Anywhere database is a file, usually with an extension of .db. Adaptive Server Anywhere includes a sample database for you to work with: this is the file asademo.db in your Adaptive Server Anywhere installation directory.

A database server The database server manages the database. No other application addresses the database file directly; they all communicate with the database server.

In the documentation, a database server is indicated as follows:



Adaptive Server Anywhere provides two versions of its database server: the personal database server and the network database server. In addition to the features of the personal server, the network database server also supports client/server communications across a network, while the personal database server can accept connections only from applications running on the same machine. The request-processing engine is identical in both servers.

A programming interface   Applications communicate with the database server using a programming interface. You can use ODBC, JDBC, OLE DB, Sybase Open Client, or Embedded SQL.

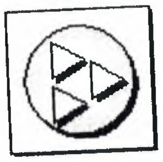
Many application development tools provide their own programming environment that hides the details of the underlying interface. For example, if you develop an application using Sybase PowerBuilder, you never have to make an ODBC function call. Nevertheless, behind the scenes each of these tools is using one of the programming interfaces.

The programming interface provides a library of function calls for communicating with the database. For ODBC and JDBC, the library is commonly called a **driver**. The library is typically provided as a shared library on UNIX operating systems or a dynamic link library (DLL) on PC operating systems. The JDBC interface uses the Sybase jConnect driver, which is a zip file of compiled Java classes.

A client application   Client applications use one of the programming interfaces to communicate with the database server.

If you develop an application using a rapid application development (RAD) tool such as Sybase PowerBuilder, you may find that the tool provides its own methods for communicating with database servers, and hides the details of the language interface. Nevertheless, all applications use one of the supported interfaces.

In diagrams in the documentation, a client application is indicated by the following icon:



UltraLite database servers are custom-generated for each UltraLite application, and are part of the application itself. An UltraLite application together with its database server is indicated as follows:

1.1.3 How the pieces fit together

Database applications can connect to a database server located on the same machine as the application itself, or in the case of the network database server, on a different machine. In addition, with Adaptive Server Anywhere you can build distributed databases, with physically distinct databases on different machines sharing data.

1.1.3.1 Personal applications and embedded databases

1.1.3.2 Client/server applications and multi-user databases

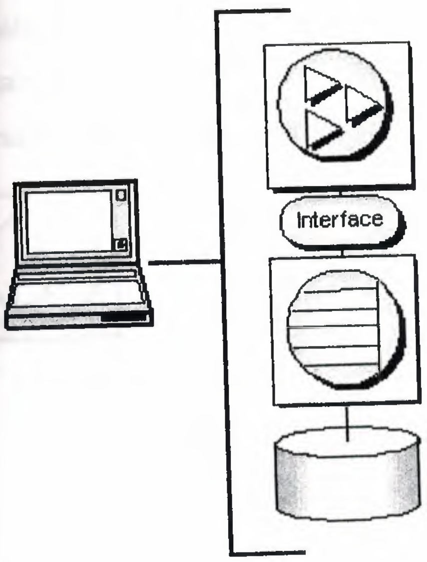
1.1.3.1 Personal applications and embedded databases

You can use Adaptive Server Anywhere to build a complete application and database on a single computer. In the simplest arrangement, this is a standalone application or personal application: it is self-contained with no connection to other databases. In this case, the database server and the database can be started by the client application, and it is common to refer to the database as an embedded database. As far as the end user is concerned, the database is a part of the application. When a database server is used as an embedded database, it is sometimes called a database engine.

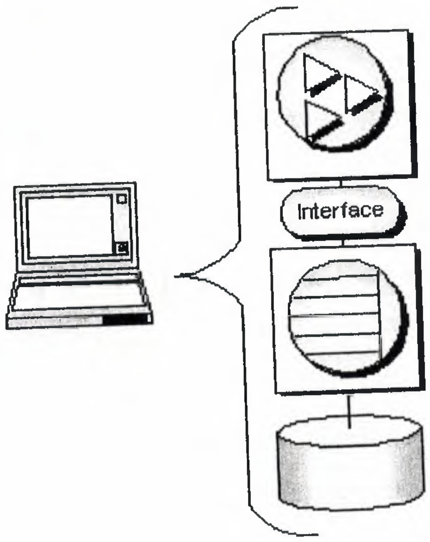
Many relational database management systems require experienced staff for administration. A characteristic of embedded databases is the ability to run entirely without administration.

The Adaptive Server Anywhere personal database server is generally used for standalone applications. A client application connects through a programming interface to a database server running on the same machine:

Standalone applications have the following architecture, with a client application connecting through a programming interface to a database server running on the same machine:



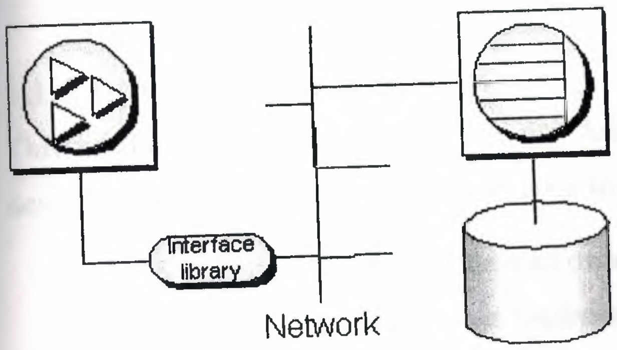
The Adaptive Server Anywhere personal database server is generally used for standalone applications, although you can also use applications on the same machine as the network server.



1.1.3.2 Client/server applications and multi-user databases

You can use Adaptive Server Anywhere to build an installation with many applications running on different machines, connected over a network to a single

database server running on a separate machine. This is a client/server or multi-user database environment, and has the following architecture. The interface library is located on each client machine.



In this case, the database server is the Adaptive Server Anywhere network database server, which supports network communications.

For a client application to work in a client/server environment, you need only identify the server to which it should connect.

For a client application to work in a client/server environment, you need only identify the server to which it should connect.

1.1.4 Multi-tier computing architecture

In multi-tier computing, application logic is held in an application server, such as Sybase EAServer, which sits between the database server and the client applications. In many situations, a single application server may access multiple databases in addition to non-relational data stores. In the Internet case, client applications are browser-based, and the application server is generally a Web server extension.

Sybase EAServer stores application logic in the form of components, and makes these components available to client applications. The components may be PowerBuilder components, Java beans, or COM components.

Application servers can also provide transaction logic to their client applications—guaranteeing that sets of operations are executed atomically across multiple databases. Adaptive Server Anywhere is well-suited to multi-tier computing, and can participate in distributed transactions coordinated by Microsoft Distributed Transaction Coordinator.

Both Sybase Enterprise Application Server and Microsoft Transaction Server use DTC to provide transaction services to their client applications.

1.1.5 Using multiple databases

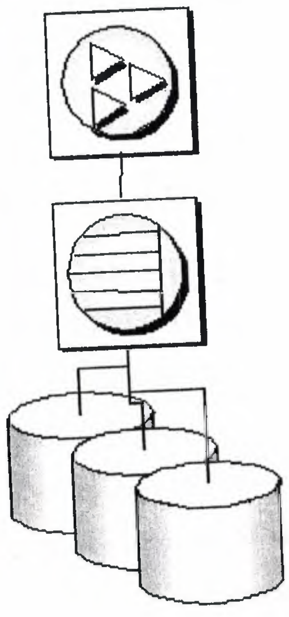
This section describes extensions to the Adaptive Server Anywhere architecture described above for the case where you want to use more than one database.

1.1.5.1 Running multiple databases on a single database server

1.1.5.2 Accessing data in other databases

1.1.5.1 Running multiple databases on a single database server

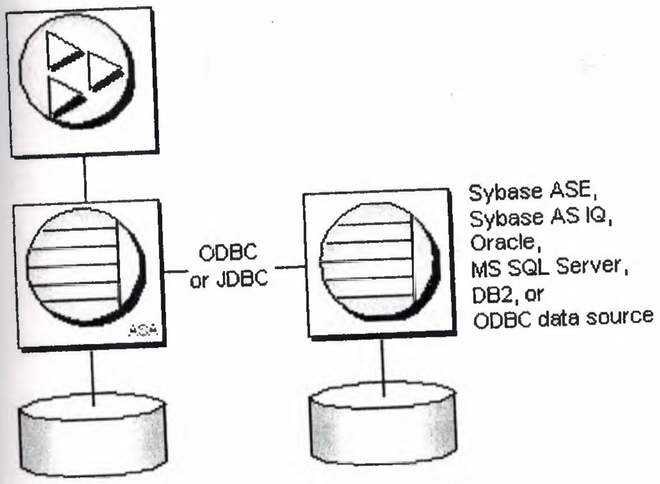
Both the Adaptive Server Anywhere personal database server and the network database server can manage several databases simultaneously. Each connection from an application must be to a single database, but an application can use separate connections to different databases, or a set of applications can work on different databases, all through the same database server.



Databases can be started when the database server is started, or by connecting to a database using the DatabaseFile connection parameter.

1.1.5.2 Accessing data in other databases

You can access databases on multiple database servers, or even on the same server, using the Adaptive Server Anywhere Remote Data Access features. The application is still connected to a single database as in the architecture diagrams above, but by defining remote servers, you can use proxy tables that exist on the remote database as if they were in the database to which you are connected.



1.2 About Dbase And Paradox

1.2.1 dBASE IV Table Specification

The dBASE IV table format was introduced in dBASE IV for DOS. Following are the specifications for dBASE IV tables.

2GB file size.

Two billion records per file.

A maximum of 255 fields per record.

Maintained indexes can have up to 47 indexes per file. Each index can be created using field expressions of virtually any combination, including conditional expressions of up to 255 characters per expression that result in an index of up to 100 bytes.

Unlimited nonmaintained indexes can be stored on disk. You can use up to 47 of them simultaneously.

1.2.2 dBase V Table Specifications

The dBASE V table format was introduced in dBASE V for Windows. Following are the specifications for dBASEA V tables.

- Up to one billion records per file.
- A maximum of 1,024 fields per record.
- Up to 32,767 bytes per record.
- Unlimited nonmaintained indexes can be stored on disk. You can use up to 47 of them simultaneously.
- Up to 10 master index files open per database. Each master index can have up to 47 indexes.
- Maintained indexes can have up to 47 indexes per file. Each index can be created using field expressions of virtually any combination, including conditional expressions of up to 255 characters per expression that result in an index of up to 100 bytes.

1.2.3 dBASE Field Types

Character (C)

dBASE III+, IV, and V field type that can contain up to 254 characters (including blank spaces). This field is similar to the Paradox Alpha field type.

Date (D)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V. dBASE tables can store dates from January 1, 100, to December 31, 9999. Paradox 5 tables can store from 12/31/9999 B.C. to 12/31/9999 A.D.

Float (F)

dBASE IV, and V floating-point numeric field type provides up to 20 significant digits.

Logical (L)

Paradox 5 and 7 and dBASE III+, IV, and V field type can store values representing True or False (yes or no). By default, valid entries include T and F (case is not important).

Memo (M)

Paradox 4, 5, and 7 as well as dBASE III+, IV, and V field. A Paradox field type is an Alpha variable-length field up to 256MB per field. dBASE Memo fields can contain binary as well as memo data.

OLE (O)

Paradox 1, 5, and 7 as well as dBASE V field type that can store OLE data.

Number (N)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V field type can store up to 15 significant digits -10307 to + 10308 with up to 15 significant digits.

dBASE number fields contain numeric data in a Binary Coded Decimal (BCD) format. Use number fields when you need to perform precise calculations on the field data. Calculations on number fields are performed more slowly but with greater precision than are calculations on float number fields. The size of a dBASE number field can be from 1 to 20. Remember, however, that BCD is in Paradox 5 and 7 only for compatibility and is mapped directly to the Number field type.

Short (S)

Paradox 3.5, 4, 5, and 7 field type that can contain integers from -- 32,767 through 32,767 (no decimal).

1.2.4 Paradox Standard Table Specifications

Also known as Paradox 4 table structure.

The Paradox standard table format was introduced in Paradox for DOS version 4. Other products that use the standard format include Paradox for DOS version 4.5, ObjectVision 2.1, and Paradox for Windows versions 1.0 and 4.5.

Earlier versions of the Paradox table type are referred to as the Compatible table type. In the BDE Configuration Utility, the level option for the Paradox driver dictates what default table type is created by Paradox for Windows. Use 3 for Compatible tables, 4 for Standard tables (the default). Following are the specifications for standard Paradox tables.

256MB file size limit if the table is in Paradox format and using a 4K block size.

Up to 255 fields per record.

Up to 64 validity checks per table.

A primary index can have up to 16 fields.

Tables can have up to 127 secondary indexes.

Up to two billion records per file. Because of the 256MB file size limit and other factors such as block size, however, the limit is much smaller. Tables of 190,000 records are easily achievable (and you can have more if you don't use up the 1,350-bytes-per-record limit for a keyed table). Tables with close to a million records are common.

Block size can be 1024, 2048, 3072, or 4096. Paradox stores data in fixed records. Even if part or all of the record is empty, the space is claimed. Knowing the interworkings can save you disk space. Paradox stores records in fixed blocks of 1024, 2048, 3072, 4096 in size.

After a block size is set for a table, that size is fixed, and all blocks in the table will be of that size. To conserve disk space, you want to try to get your record size as close to a multiple of block size as possible (minus 6 bytes, which are used by Paradox to manage the table).

Record size. 1,350 for keyed tables and 4,000 for unkeyed tables. When figuring out the size (the number of bytes or characters) of a table, remember that Alpha fields take up their size (for example, an A10 = 10 bytes), numeric field types take up 8 bytes, short number field types take up 2 bytes, money takes up 8, and dates take up 4 bytes.

Memos, BLOBs, and so on take 10 bytes plus however much of the memo is stored in the .DB. For example, M15 takes 25 bytes.

1.2.5 Paradox 5 Table Specifications

The Paradox 5 table format was introduced in Paradox for Windows version 5. Following are the specifications for Paradox 5 tables.

Up to two billion records per file.

File size is limited to two gigabytes.

Up to 255 fields per record.

Record size: Up to 10,800 bytes per record for indexed tables and 32,750 bytes per record for nonindexed tables. When figuring out the size (the number of bytes or characters) of a table, remember that Alpha fields take up their size (for example, an

A10 = 10 bytes), numeric field types take up 8 bytes, short number field types take up 2 bytes, money takes up 8, and dates take up 4 bytes.

Memos, BLOBs, and so on take 10 bytes plus however much of the memo is stored in the .DB. For example, M15 takes 25 bytes.

Up to 64 validity checks per table for Paradox for Windows tables.

A primary index can have up to 16 fields.

Tables can have up to 127 secondary indexes.

Block size can be from 1K to 32K in steps of 1K. For example, 1024, 2048, 3072, 4096, 5120...32768.

1.2.6 Paradox 7 and above Table Specifications

The Paradox 7 table format was introduced in Paradox version 7 for Windows 95/NT. The Paradox 7 table format has all the same specifications as the Paradox 5 table format with two additions. Following are the specification additions for the Paradox 7 table format.

- Added descending secondary indexes.
- Added unique secondary indexes

1.2.7 Paradox Field Types

Alpha (A)

Paradox 3.5, 4, 5, and 7 field type that can contain up to 255 letters and numbers. This field type was called Alphanumeric in versions of Paradox before version 5. It is similar to the Character field type in dBASE.

Autoincrement (+)

Field type introduced in the Paradox 5 table format that adds one to the highest number in the table whenever a record is inserted. The starting range can from -2,147,483,647 to 2,147,483,647. Deleting a record does not change the field values of other records.

BCD (#)

Paradox 5 and 7 field type which is provided only for compatibility with other applications that use BCD data. Paradox correctly interprets BCD data from other applications that use the BCD type. When Paradox performs calculations on BCD data, it converts the data to the numeric float type, then converts the result back to BCD. When this field type is fully supported, it will support up to 32 significant digits.

Binary (B)

Paradox 1, 5, and 7 field type that can store binary data up to 256MB per field.

Bytes (Y)

Paradox 5 and 7 field type for storing binary data up to 255 bytes. Unlike binary fields, bytes fields are stored in the Paradox table (rather than in the separate .MB file), allowing for faster access.

Date (D)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V. dBASE tables can store dates from January 1, 100, to December 31, 9999. Paradox 5 tables can store from 12/31/9999 B.C. to 12/31/9999 A.D.

Formatted Memo (F)

Paradox 1, 4.5, 5, and 7 field type is like a memo field except that you can format the text. You can alter and store the text attributes of typeface, style, color, and size. This rich text document has a variable-length up to 256MB per field.

Graphic (G)

Paradox 1, 5, and 7 field type can contain pictures in .BMP (up to 24 bit), .TIF (up to 256 color), .GIF (up to 256 color), .PCX, and .EPS file formats. Not all graphic variations are available. For example, currently you cannot store a 24-bit .TIF graphic. When you paste a graphic into a graphic field, Paradox converts the graphic into the .BMP format.

Logical (L)

Paradox 5 and 7 and dBASE III+, IV, and V field type can store values representing True or False (yes or no). By default, valid entries include T and F (case is not important).

Memo (M)

Paradox 4, 5, and 7 as well as dBASE III+, IV, and V field. A Paradox field type is an Alpha variable-length field up to 256MB per field. dBASE Memo fields can contain binary as well as memo data.

For Paradox tables, the file is divided into blocks of 512 characters. Each block is referenced by a sequential number, beginning at zero. Block 0 begins with a 4-byte number in hexadecimal format, in which the least significant byte comes first. This number specifies the number of the next available block. It is, in effect, a pointer to the end of the memo file. The remainder of Block 0 isn't used.

Money ($)

Paradox 3.5, 4, 5, and 7 field type, like number fields, can contain only numbers. They can hold positive or negative values. Paradox recognizes up to six decimal places when performing internal calculations on money fields. This field type was called Currency in previous versions of Paradox.

OLE (O)

Paradox 1, 5, and 7 as well as dBASE V field type that can store OLE data.

Number (N)

Paradox 3.5, 4, 5, and 7 as well as dBASE III+, IV, and V field type can store up to 15 significant digits -10307 to + 10308 with up to 15 significant digits.

dBASE number fields contain numeric data in a Binary Coded Decimal (BCD) format. Use number fields when you need to perform precise calculations on the field data. Calculations on number fields are performed more slowly but with greater precision than are calculations on float number fields. The size of a dBASE number field can be from 1 to 20. Remember, however, that BCD is in Paradox 5 and 7 only for compatibility and is mapped directly to the Number field type.

Short (S)

Paradox 3.5, 4, 5, and 7 field type that can contain integers from -- 32,767 through 32,767 (no decimal).

Time (T)

Paradox 5 and 7 field type that can contain time times of day, stored in milliseconds since midnight and limited to 24 hours.

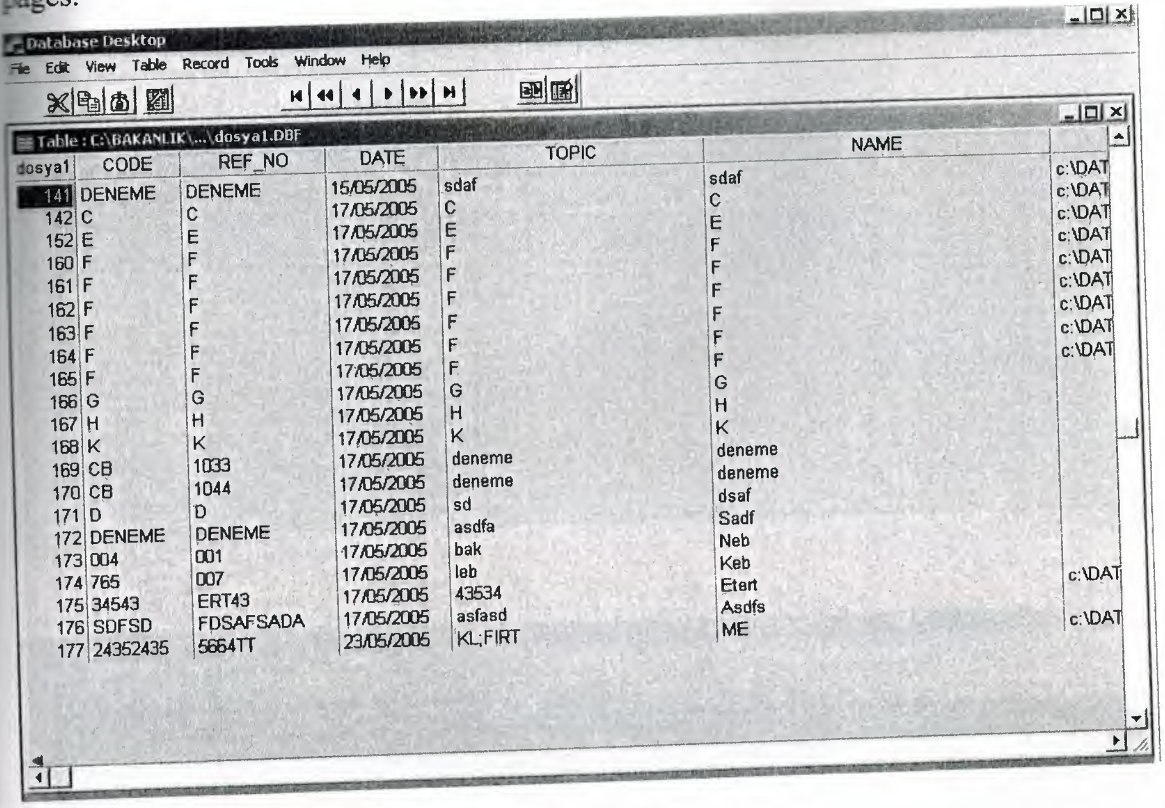
This field type does not store duration which is the difference between two times. For example, if you need to store the duration of a song, use an Alpha field. Whenever you need to store time, make a distinction between clock time and duration. The Time field type is perfect for clock time. Duration can be stored in an Alpha field and manipulated with code.

TimeStamp (@)

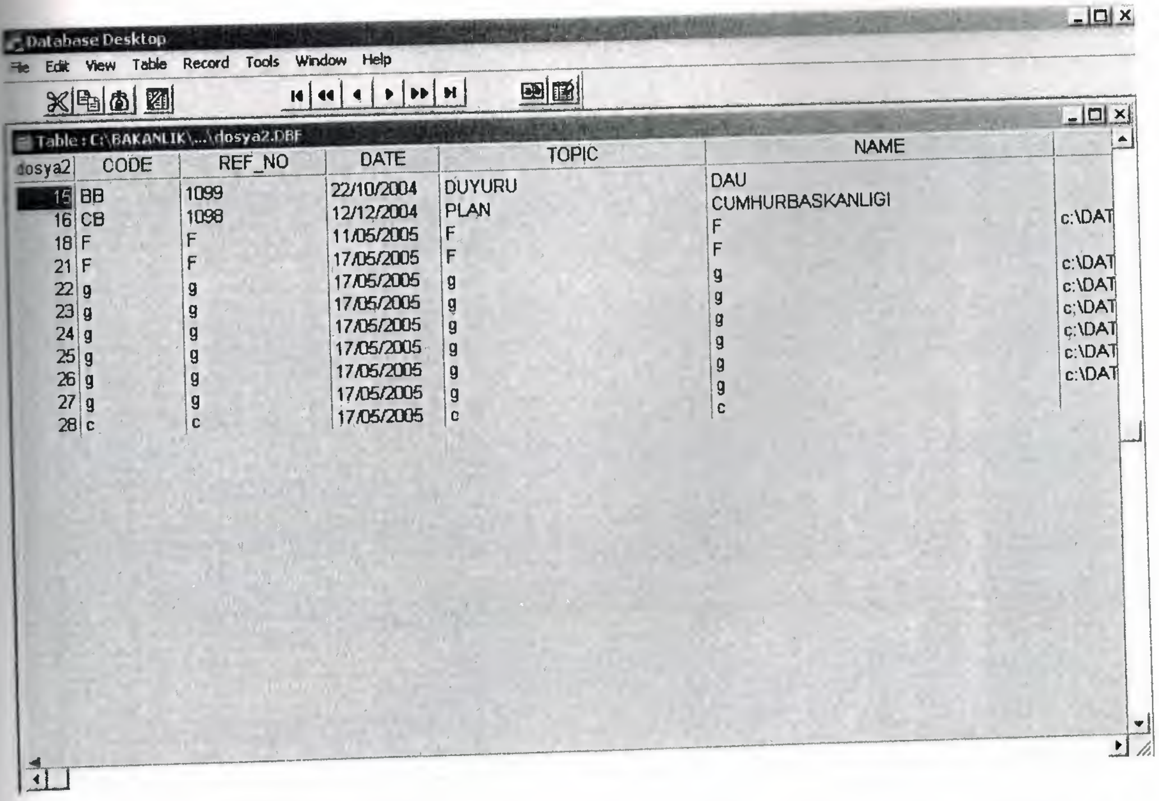
Paradox 5 field type comprised of both date and time values. Rules for this field type are the same as those for date fields and time fields.

1.2.8 About Our Database

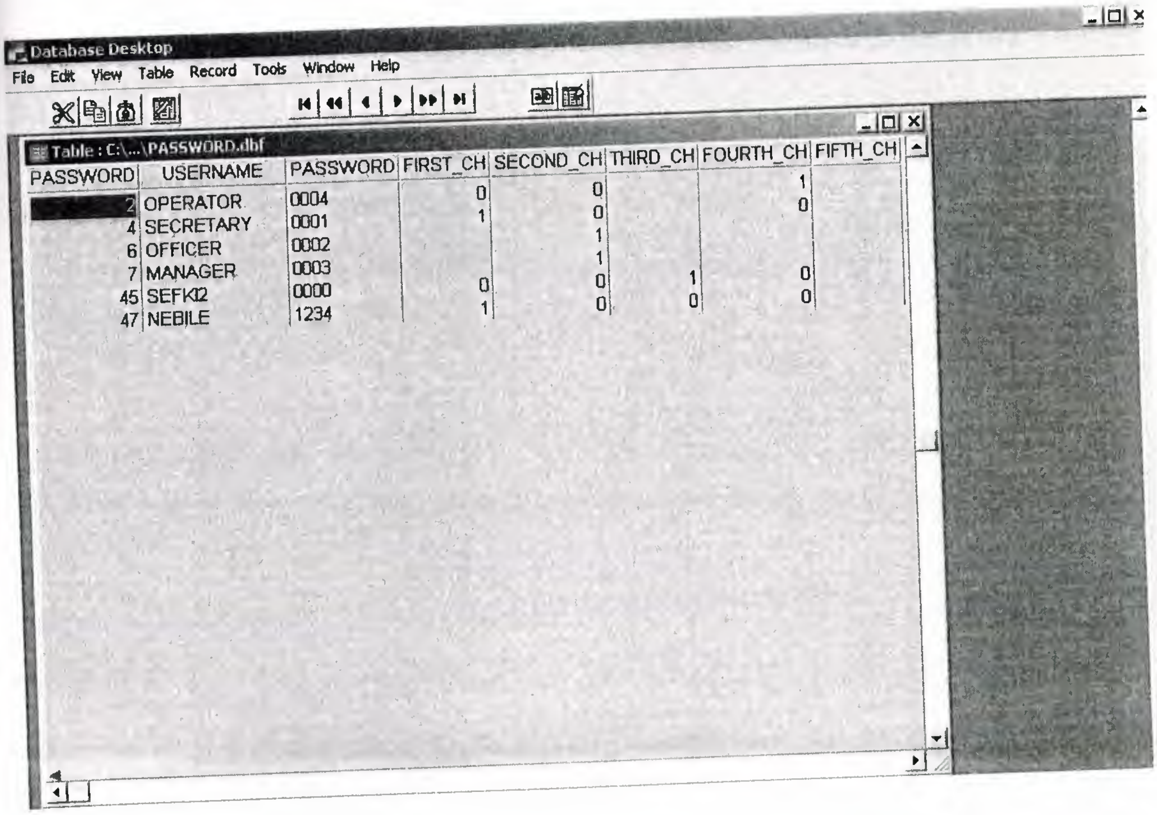
In our archive program we used dbase for database .You can see the tables in following pages.


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\BAKANLIK\...\dosya1.DBF

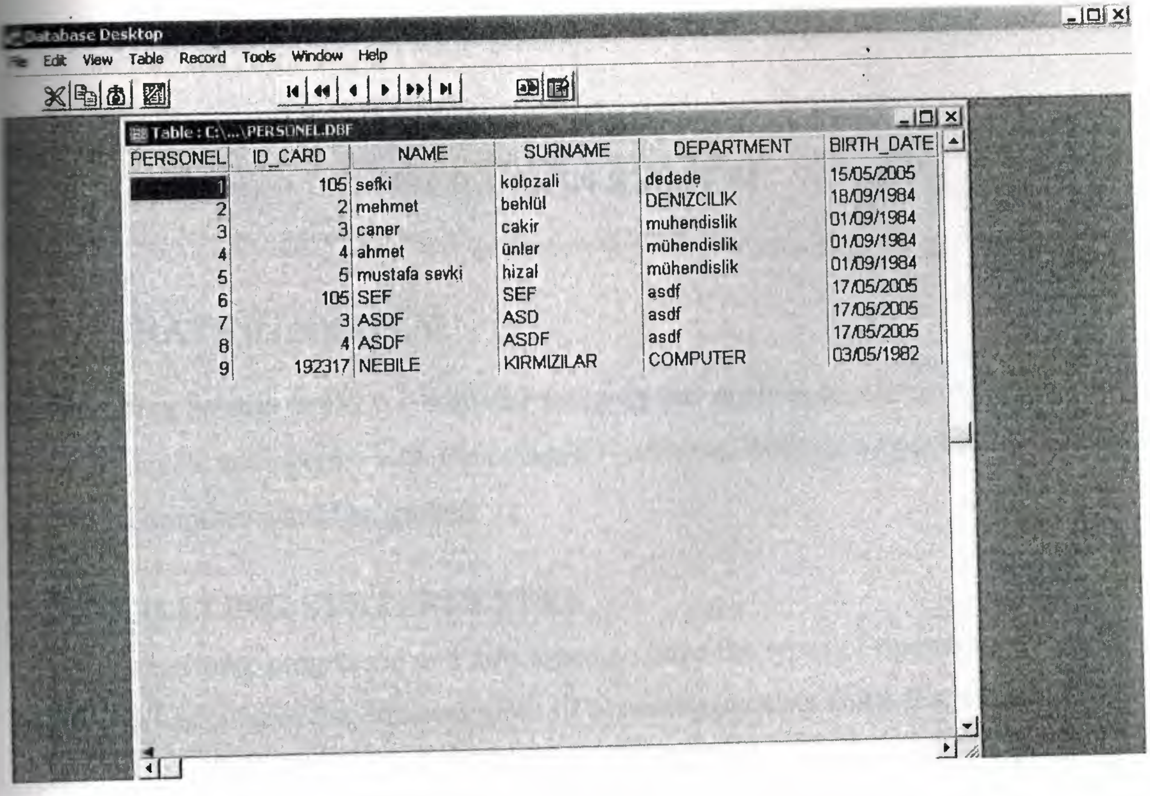
dosya1	CODE	REF_NO	DATE	TOPIC	NAME	
141	DENEME	DENEME	15/05/2005	sdaf	sdaf	c:\DAT
142	C	C	17/05/2005	C	C	c:\DAT
152	E	E	17/05/2005	E	E	c:\DAT
160	F	F	17/05/2005	F	F	c:\DAT
161	F	F	17/05/2005	F	F	c:\DAT
162	F	F	17/05/2005	F	F	c:\DAT
163	F	F	17/05/2005	F	F	c:\DAT
164	F	F	17/05/2005	F	F	c:\DAT
165	F	F	17/05/2005	F	F	c:\DAT
166	G	G	17/05/2005	G	G	
167	H	H	17/05/2005	H	H	
168	K	K	17/05/2005	K	K	
169	CB	1033	17/05/2005	deneme	deneme	
170	CB	1044	17/05/2005	deneme	deneme	
171	D	D	17/05/2005	sd	dsaf	
172	DENEME	DENEME	17/05/2005	asdfa	Sadf	
173	004	001	17/05/2005	bak	Neb	
174	765	007	17/05/2005	leb	Keb	
175	34543	ERT43	17/05/2005	43534	Etert	c:\DAT
176	SDFSD	FDSAFSADA	17/05/2005	asfasd	Asdfs	
177	24352435	5664TT	23/05/2005	KL;FIRT	ME	c:\DAT


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\BAKANLIK\...\dosya2.DBF

dosya2	CODE	REF_NO	DATE	TOPIC	NAME	
15	BB	1099	22/10/2004	DUYURU	DAU	
16	CB	1098	12/12/2004	PLAN	CUMHURBASKANLIGI	
18	F	F	11/05/2005	F	F	c:\DAT
21	F	F	17/05/2005	F	F	
22	g	g	17/05/2005	g	g	c:\DAT
23	g	g	17/05/2005	g	g	c:\DAT
24	g	g	17/05/2005	g	g	c;\DAT
25	g	g	17/05/2005	g	g	c:\DAT
26	g	g	17/05/2005	g	g	c:\DAT
27	g	g	17/05/2005	g	g	c:\DAT
28	c	c	17/05/2005	c	c	


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\...\PASSWORD.dbf

PASSWORD	USERNAME	PASSWORD	FIRST_CH	SECOND_CH	THIRD_CH	FOURTH_CH	FIFTH_CH
2	OPERATOR	0004	0	0		1	
4	SECRETARY	0001	1	0		0	
6	OFFICER	0002		1			
7	MANAGER	0003		1			
45	SEFKI2	0000	0	0	1	0	
47	NEBILE	1234	1	0	0	0	


Database Desktop
File Edit View Table Record Tools Window Help
Table : C:\...\PERSONEL.DBF

PERSONEL	ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
1	105	sefki	kolozali	dedede	15/05/2005
2	2	mehmet	behlül	DENIZCILIK	18/09/1984
3	3	caner	cakir	muhendislik	01/09/1984
4	4	ahmet	ünler	mühendislik	01/09/1984
5	5	mustafa sevki	hizal	mühendislik	01/09/1984
6	105	SEF	SEF	asdf	17/05/2005
7	3	ASDF	ASD	asdf	17/05/2005
8	4	ASDF	ASDF	asdf	17/05/2005
9	192317	NEBILE	KIRMIZILAR	COMPUTER	03/05/1982

CHAPTER 2

OPERATION SYSTEM

2.1 OPERATING SYSTEM

An Operating System or OS is a software program that enables the computer hardware to communicate and operate with the computer software. Without a computer Operating System a computer would be useless.

2.2 OPERATING SYSTEM TYPES

As computers have progressed and developed so have the types of operating systems. Below is a basic list of the different types of operating systems and a few examples of Operating Systems that fall into each of the categories. Many computer Operating Systems will fall into more then one of the below categories.

2.2.1 Multi-user

A multi-user Operating System allows for multiple users to use the same computer at the same time and/or different times.

Below are some examples of multi-user Operating Systems.

Linux
UNIX
Windows 2000

2.2.2 Multiprocessing

An Operating System capable of supporting and utilizing more than one computer processor.

1. When referring to a network, a multi-user system is a term commonly used to define a computer capable of allowing multiple users to connect to a network.

2. When referring to a computer Operating System, a multi-user system is a computer with an Operating system that supports multiple users at once and/or different times.

Below are some examples of multiprocessing Operating Systems.

Linux
UNIX
Windows 2000

2.2.3 Multitasking

An Operating systems that is capable of allowing multiple software processes to be run at the same time. Below are some examples of multitasking Operating Systems.

UNIX
Windows 2000

2.2.4 Multithreading

Operating systems that allow different parts of a software program to run concurrently. Operating systems that would fall into this category are:

Linux
UNIX
Windows 2000

2.3 Introduction to UNIX:

2.3.1 Objectives

Its covers:

- The concept of an operating system.
- The internal architecture of an operating system.
- The evolution of the UNIX operating system into two broad schools (BSD and SYSV) and the development of Linux, a popular open source operating system.

- The architecture of the Linux operating system in more detail.
- How to log into (and out of) UNIX and change your password.
- The general format of UNIX commands.

2.3.2 What is an Operating System?

An operating system (OS) is a resource manager. It takes the form of a set of software routines that allow users and application programs to access system resources (e.g. the CPU, memory, disks, modems, printers network cards etc.) in a safe, efficient and abstract way.

For example, an OS ensures safe access to a printer by allowing only one application program to send data directly to the printer at any one time. An OS encourages efficient use of the CPU by suspending programs that are waiting for I/O operations to complete to make way for programs that can use the CPU more productively. An OS also provides convenient abstractions (such as files rather than disk locations) which isolate application programmers and users from the details of the underlying hardware.

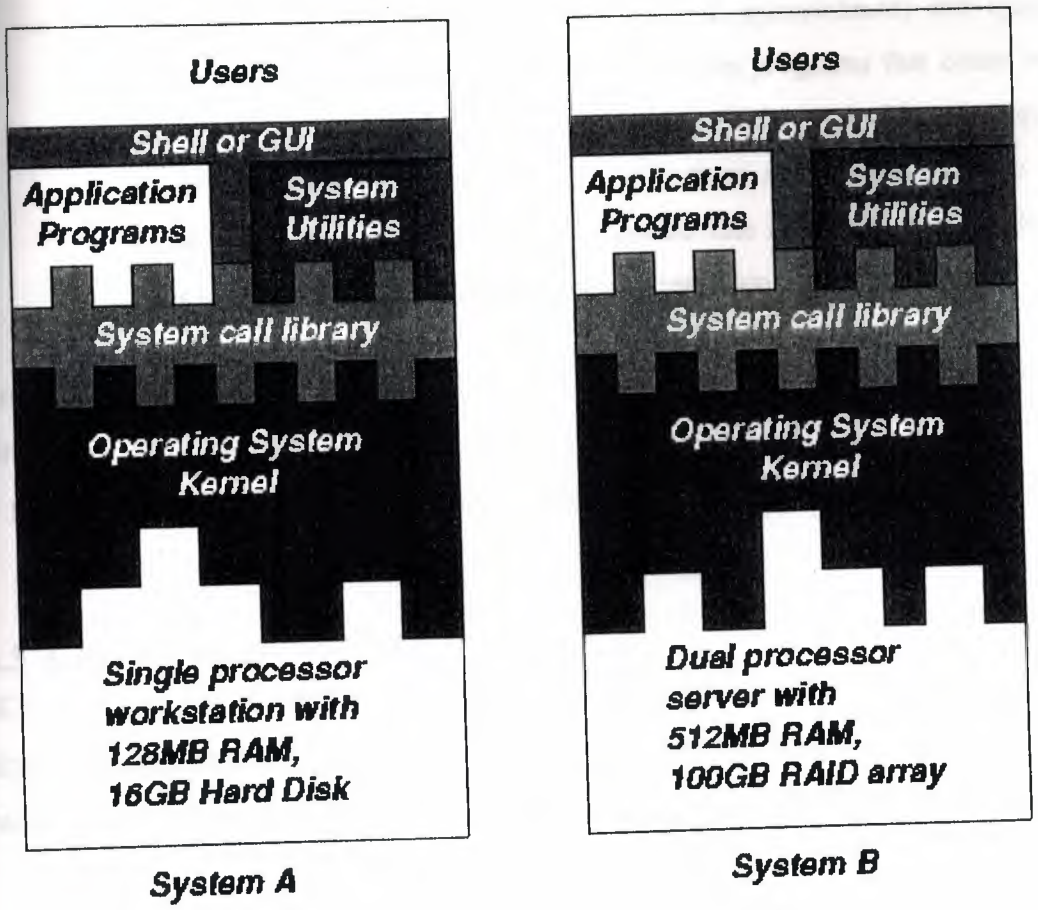


Fig. 1.1: General operating system architecture

Fig. 1.1 presents the architecture of a typical operating system and shows how an OS succeeds in presenting users and application programs with a uniform interface without regard to the details of the underlying hardware. We see that:

- The operating system kernel is in direct control of the underlying hardware. The kernel provides low-level device, memory and processor management functions (e.g. dealing with interrupts from hardware devices, sharing the processor among multiple programs, allocating memory for programs etc.)
- Basic hardware-independent kernel services are exposed to higher-level programs through a library of system calls (e.g. services to create a file, begin execution of a program, or open a logical network connection to another computer).

- Application programs (e.g. word processors, spreadsheets) and system utility programs (simple but useful application programs that come with the operating system, e.g. programs which find text inside a group of files) make use of system calls. Applications and system utilities are launched using a shell (a textual command line interface) or a graphical user interface that provides direct user interaction.

Operating systems (and different flavours of the same operating system) can be distinguished from one another by the system calls, system utilities and user interface they provide, as well as by the resource scheduling policies implemented by the kernel.

2.3.3 A Brief History of UNIX

UNIX has been a popular OS for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities. What follows here is a simplified history of how UNIX has developed.

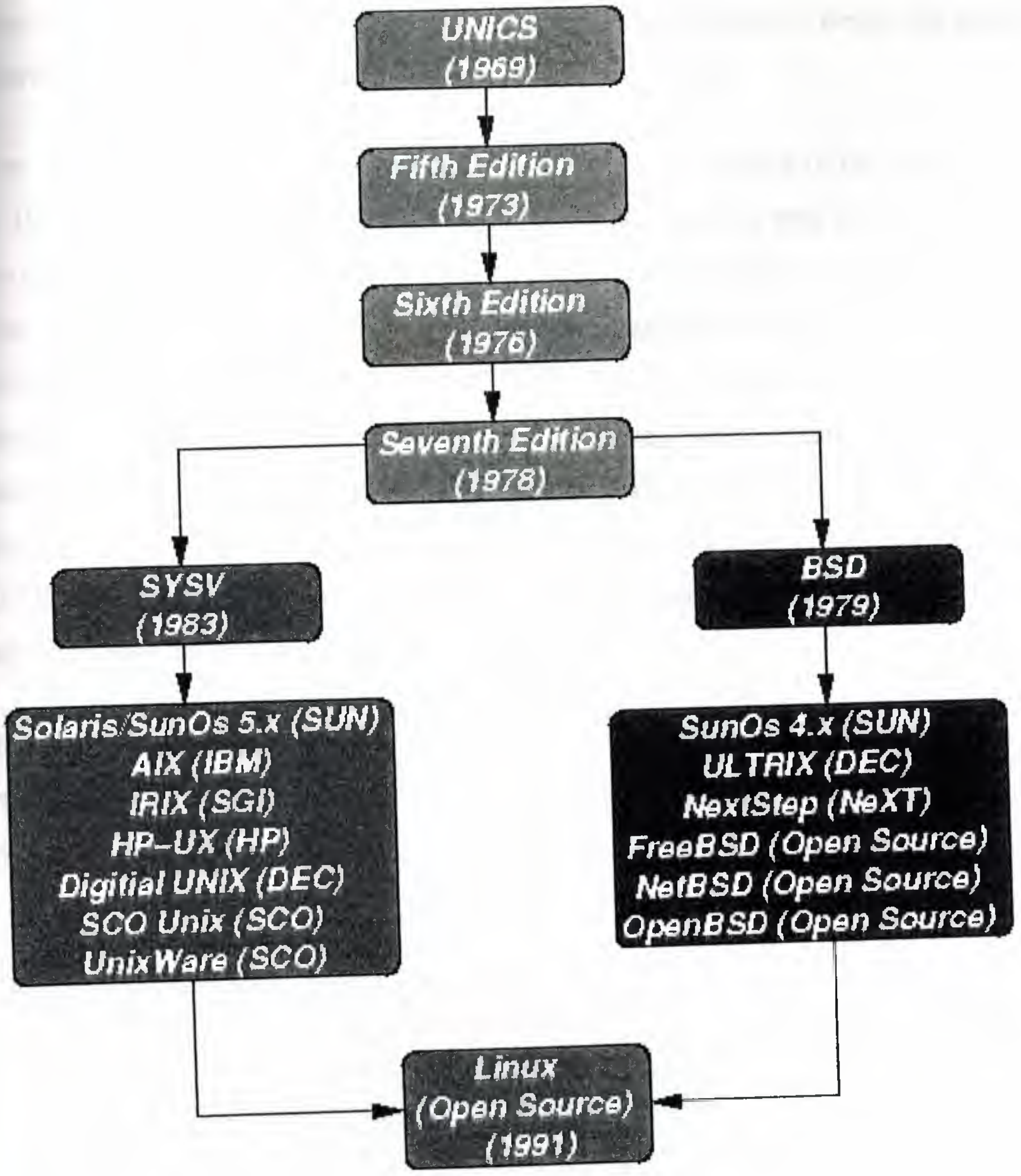


Fig. 1.2: Simplified UNIX FamilyTree

In the late 1960s, researchers from General Electric, MIT and Bell Labs launched a joint project to develop an ambitious multi-user, multi-tasking OS for mainframe computers known as MULTICS (Multiplexed Information and Computing System). MULTICS failed (for some MULTICS enthusiasts "failed" is perhaps too strong a word to use here), but it did inspire Ken Thompson, who was a researcher at Bell Labs, to have a go at writing a simpler operating system himself. He wrote a simpler version of MULTICS on a PDP7 in assembler and called his attempt UNICS (Uniplexed Information and Computing System). Because memory and CPU power were at a premium in those days, UNICS (eventually shortened to UNIX) used short commands to minimize the

space needed to store them and the time needed to decode them - hence the tradition of short UNIX commands we use today, e.g. ls, cp, rm, mv etc.

Ken Thompson then teamed up with Dennis Ritchie, the author of the first C compiler in 1973. They rewrote the UNIX kernel in C - this was a big step forwards in terms of the system's portability - and released the Fifth Edition of UNIX to universities in 1974. The Seventh Edition, released in 1978, marked a split in UNIX development into two main branches: SYSV (System 5) and BSD (Berkeley Software Distribution). BSD arose from the University of California at Berkeley where Ken Thompson spent a sabbatical year. Its development was continued by students at Berkeley and other research institutions. SYSV was developed by AT&T and other commercial companies. UNIX flavours based on SYSV have traditionally been more conservative, but better supported than BSD-based flavours.

The latest incarnations of SYSV (SVR4 or System 5 Release 4) and BSD Unix are actually very similar. Some minor differences are to be found in file system structure, system utility names and options and system call libraries as shown in Fig 1.3.

Feature	**Typical SYSV**	**Typical BSD**
kernel name	/unix	/vmunix
boot init	/etc/rc.d directories	/etc/rc.* files
mounted FS	/etc/mnttab	/etc/mtab
default shell	sh, ksh	csh, tcsh
FS block size	512 bytes->2K	4K->8K
print subsystem	lp, lpstat, cancel	lpr, lpq, lprm
echo command (no new line)	echo "\c"	echo -n
ps command	ps -fae	ps -aux
multiple wait syscalls	poll	select
memory access syscalls	memset, memcpy	bzero, bcopy

Fig. 1.3: Differences between SYSV and BSD

Linux is a free open source UNIX OS for PCs that was originally developed in 1991 by Linus Torvalds, a Finnish undergraduate student. Linux is neither pure SYSV or pure BSD. Instead, incorporates some features from each (e.g. SYSV-style startup files but BSD-style file system layout) and aims to conform with a set of IEEE standards called POSIX (Portable Operating System Interface). To maximise code portability, it typically supports SYSV, BSD and POSIX system calls (e.g. poll, select, memset, memcpy, bzero and bcopy are all supported).

The open source nature of Linux means that the source code for the Linux kernel is freely available so that anyone can add features and correct deficiencies. This approach has been very successful and what started as one person's project has now turned into a collaboration of hundreds of volunteer developers from around the globe. The open source approach has not just successfully been applied to kernel code, but also to application programs for Linux.

As Linux has become more popular, several different development streams or distributions have emerged, e.g. Redhat, Slackware, Mandrake, Debian, and Caldera. A distribution comprises a prepackaged kernel, system utilities, GUI interfaces and application programs.

Redhat is the most popular distribution because it has been ported to a large number of hardware platforms (including Intel, Alpha, and SPARC), it is easy to use and install and it comes with a comprehensive set of utilities and applications including the X Windows graphics system, GNOME and KDE GUI environments, and the StarOffice suite (an open source MS-Office clone for Linux).

2.3.4 Architecture of the Linux Operating System

Linux has all of the components of a typical OS (at this point you might like to refer back to Fig 1.1):

- **Kernel**

  The Linux kernel includes device driver support for a large number of PC hardware devices (graphics cards, network cards, hard disks etc.), advanced processor and memory management features, and support for many different types of filesystems (including DOS floppies and the

ISO9660 standard for CDROMs). In terms of the services that it provides to application programs and system utilities, the kernel implements most BSD and SYSV system calls, as well as the system calls described in the POSIX.1 specification.

The kernel (in raw binary form that is loaded directly into memory at system startup time) is typically found in the file /boot/vmlinuz, while the source files can usually be found in /usr/src/linux.The latest version of the Linux kernel sources can be downloaded from http://www.kernel.org.

- **Shells and GUIs**
  Linux supports two forms of command input: through textual command line shells similar to those found on most UNIX systems (e.g. sh - the Bourne shell, bash - the Bourne again shell and csh - the C shell) and through graphical interfaces (GUIs) such as the KDE and GNOME window managers. If you are connecting remotely to a server your access will typically be through a command line shell.

- **System Utilities**
  Virtually every system utility that you would expect to find on standard implementations of UNIX (including every system utility described in the POSIX.2 specification) has been ported to Linux. This includes commands such as ls, cp, grep, awk, sed, bc, wc, more, and so on. These system utilities are designed to be powerful tools that do a single task extremely well (e.g. grep finds text inside files while wc counts the number of words, lines and bytes inside a file). Users can often solve problems by interconnecting these tools instead of writing a large monolithic application program.

Like other UNIX flavours, Linux's system utilities also include server programs called daemons which provide remote network and administration services (e.g. telnetd and sshd provide remote login facilities, lpd provides printing services, httpd serves web

pages, crond runs regular system administration tasks automatically). A daemon (probably derived from the Latin word which refers to a beneficient spirit who watches over someone, or perhaps short for "Disk And Execution MONitor") is usually spawned automatically at system startup and spends most of its time lying dormant (lurking?) waiting for some event to occur.

- **Application programs**

  Linux distributions typically come with several useful application programs as standard. Examples include the emacs editor, xv (an image viewer), gcc (a C compiler), g++ (a C++ compiler), xfig (a drawing package), latex (a powerful typesetting language) and soffice (StarOffice, which is an MS-Office style clone that can read and write Word, Excel and PowerPoint files).

  Redhat Linux also comes with rpm, the Redhat Package Manager which makes it easy to install and uninstall application programs.

2.4 An Introduction to Linux

2.4.1 What is Linux?

Linux is a UNIX-based operating system originally developed as for Intel-compatible PC's. It is now available for most types of hardware platforms, ranging from PDAs (and according to some reports, a wristwatch) to mainframes. Linux is a "modern operating system", meaning it has such features as virtual memory, memory protection, and preemptive multitasking.

Linux is built and supported by a large international community of developers and users dedicated to free, open-source software. This community sees Linux as an alternative to such proprietary systems as Windows and Solaris, and as a platform for alternatives to such proprietary applications as MS Office, Internet Explorer, and Outlook.

As a result of this community, there is a very large collection of free software available for Linux. There are graphical environments (GUIs), office applications, developers'

tools, system utilities, business applications, document publishing tools, network client and server applications -- the list goes on.

The best part of this community is that all code is open. This means there is no barrier to entry; for any given problem, there are generally several applications that solve the problem. These applications can also borrow the best parts from each other to become even better. An excellent example of this is Galeon. Galeon is a web browser which took Mozilla's web page rendering engine and integrated it with a GTK frontend (instead of Mozilla's normal frontend).

Linux specifically refers to the Linux kernel. However, the kernel is useless without a set of tools and applications to run on the kernel. Linux is most commonly distributed with this toolset and a collection of applications in what is called a "distribution". The most common are Redhat, Mandrake, Suse, and Debian. Distributions differ in three basic ways: the process for installing the distribution, the applications available, and process for installing and managing these applications.

2.4.2 Why use Linux?

Reasons to Install Linux

- Configurability
- Convenience
- Stability
- Community
- Freedom

2.4.2.1 Configurability

Linux distributions give the user full access to configure just about any aspect of their system. Options range from the simple and straightforward (for instance, changing the background image) to the more esoteric (for instance, making the "Caps Lock" key behave like "Control"). Almost any aspect of the user experience can be configured.

Linux also allows the user to automate just about any task. Advanced scripting and high-level programming are standard features. Most operations are accessible via these scripting options. Finally, Linux offers the ultimate in configurability: the source code, to be modified as you see fit.

2.4.2.2 Convenience

While Linux takes some effort to get set up, once it is set up, it is surprisingly low-maintenance. Package management can simply be a matter of running two commands in the shell. Linux also offers complete remote access. This allows the user to act exactly as if she is sitting at that computer's desk, potentially across town or on the other side of the world.

2.4.2.3 Stability

Linux is based on the UNIX kernel. It provides preemptive multitasking and protected memory. Preemptive multitasking prevents any application from permanently stealing the CPU and locking up the machine. Protected memory prevents applications from interfering with and crashing one-another.

Linux and related tools are also open-source. This means that the source code is available for the public to view. There are literally hundreds, if not thousands, of developers working on the various pieces of Linux. In this open development process, bugs are fixed very quickly. In addition, bugs are fixed immediately, instead of waiting for the next major release. It certainly helps that the people who develop Linux and associated tools use their programs every day.

2.4.2.4 Community

Linux is part of the greater open-source community. This consists of thousands of developers and many more users world-wide who support open software. This user and developer base is also a support base.

In Rice, there is the Rice Linux Users Group (the group who are bringing you this class). We hold regular meetings where people can bring up their Linux problems. There is also the newsgroup rice.comp.linux, where questions can be asked or problems laid out any time, day or night. This newsgroup is mirrored to the RLUG-discuss mailing list, for RLUG members who don't have access to Rice newsgroups.

Worldwide, the Linux community is even greater. There is a mailing list for just about every project or piece of software in active development -- if you have a question about

a program, who better to ask than the person who wrote it? There are also newsgroups and web pages which have collectively with the mailing lists probably addressed every problem someone new to Linux has encountered several times over.

2.4.2.5 Freedom

Linux is free. This means more than just costing nothing. This means that you are allowed to do whatever you want to with the software. This is why Redhat, Mandrake, and Suse are all allowed to sell their own distributions of Linux. The only restriction placed on Linux is that, if you distribute Linux, you must grant all the privileges to the code that you had, including providing the source. This prevents a corporation from using the Linux kernel as the basis for their proprietary operating system.

2.5 The differences between Linux and other UNIX

The differences between Linux and other UNIX operating systems is barely noticable to the average user. Commands might be located in slightly different places or may have different sets of arguments but the functionallity is the same. With the GNU tools available for most platforms these days even these differences are starting to vanish. Most of the differences are located in the kernel design and the systems administration tools. Some systems such as AIX and HP-UX provide fancy graphical management programs while others such as Linux and Solaris tend to rely on editing files and running commands manually. To sum up, UNIX is like a car: once you can drive one, you can drive them all. However if you open up the bonnet to tinker around you'll find many diverse systems as you move from one type to another.

CHAPTER 3

DELPHI

3.1.INTRODUCTION

The name "Delphi" was never a term with which either Olaf Helmer or Norman Dalkey (the founders of the method) were particular happy. Since many of the early Delphi studies focused on utilizing the technique to make forecasts of future occurrences, the name was first applied by some others at Rand as a joke. However, the name stuck. The resulting image of a priestess, sitting on a stool over a crack in the earth, inhaling sulfur fumes, and making vague and jumbled statements that could be interpreted in many different ways, did not exactly inspire confidence in the method.

The straightforward nature of utilizing an iterative survey to gather information "sounds" so easy to do that many people have done "one" Delphi, but never a second. Since the name gives no obvious insight into the method and since the number of unsuccessful Delphi studies probably exceeds the successful ones, there has been a long history of diverse definitions and opinions about the method. Some of these misconceptions are expressed in statements such as the following that one finds in the literature:

It is a method for predicting future events.

It is a method for generating a quick consensus by a group.

It is the use of a survey to collect information.

It is the use of anonymity on the part of the participants.

It is the use of voting to reduce the need for long discussions.

It is a method for quantifying human judgement in a group setting.

Some of these statements are sometimes true; a few (e.g. consensus) are actually contrary to the purpose of a Delphi. Delphi is a communication structure aimed at

producing detailed critical examination and discussion, not at forcing a quick compromise. Certainly quantification is a property, but only to serve the goal of quickly identifying agreement and disagreement in order to focus attention. It is often very common, even today, for people to come to a view of the Delphi method that reflects a particular application with which they are familiar. In 1975 Linstone and Turoff proposed a view of the Delphi method that they felt best summarized both the technique and its objective:

"Delphi may be characterized as a method for structuring a group communication process, so that the process is effective in allowing a group of individuals, as a whole, to deal with complex problems." (page 3)

The essence of Delphi is structuring of the group communication process. Given that there had been much earlier work on how to facilitate and structure face-to-face meetings, the other important distinction was that Delphi was commonly applied utilizing a paper and pencil communication process among groups in which the members were dispersed in space and time. Also, Delphis were commonly applied to groups of a size (30 to 100 individuals) that could not function well in a face-to-face environment, even if they could find a time when they all could get together.

Additional opportunity has been added by the introduction of Computer Mediated Communication Systems (Hiltz and Turoff, 1978; Rice and Associates, 1984; Turoff, 1989; Turoff, 1991). These are computer systems that support group communications in either a synchronous (Group Decision Support Systems, Desanctis et. al., 1987) or an asynchronous manner (Computer Conferencing). Techniques that were developed and refined in the evolution of the Delphi Method (e.g. anonymity, voting) have been incorporated as basic facilities or tools in many of these computer based systems. As a result, any of these systems can be used to carry out some form of a Delphi process or Nominal Group Technique (Delbecq, et. al., 1975).

The result, however, is not merely confusion due to different names to describe the same things; but a basic lack of knowledge by many people working in these areas as to what was learned in the studies of the Delphi Method about how to properly employ these techniques and their impact on the communication process. There seems to be a great deal of "rediscovery" and repeating of earlier misconceptions and difficulties.

Given this situation, the primary objective of this chapter is to review the specific properties and methods employed in the design and execution of Delphi Exercises and to examine how they may best be translated into a computer based environment.

3.2. ASYNCHRONOUS INTERACTION

Perhaps the most important and least understood property of the Delphi method is the ability of members of a group to participate in an asynchronous manner. This property of asynchronous interaction has two characteristics:

A person may choose to participate in the group communication process when they feel they want to.

A person may choose to contribute to that aspect of the problem to which they feel best able to contribute.

It does not matter what time of the day or night Delphi participants think of good ideas to include in their response. They can fill out a Delphi survey when they wish to, or they can go to a computer terminal to contribute when they wish to. This can be done at whatever point in time the individual feels he or she has thought of significant things to include in response to the issues involved. Participants can revise and add to their responses over time, before sending them to the group monitor for dissemination to the others.

A good Delphi survey attempts to tackle the problem from many different perspectives. Sometimes this is referred to as including questions in the Delphi survey which approach the problem both from the "bottom-up" and from the "top-down" perspectives. This allows different individuals in the group to focus on the approach to problem solving with which they feel most comfortable.

In a normal face-to-face group process, and in the environment characterized by face-to-face Group Decision Support Systems, all the members of the group are forced into a lockstep treatment of a problem. When the group is considering the subject of "goals," those who have difficulty dealing with "abstraction" may feel at a disadvantage, because they do not have as much to contribute. Conversely, when focusing on specific solution approaches, those who deal better with "abstraction" may not feel they are contributing. One of the specific advantages of groups is to allow individuals with differing perspectives and/or differing cognitive abilities to contribute to those parts of a complex

problem for which they have both the appropriate knowledge and appropriate problem solving skills. A typical model for a group problem solving process is:

Recognition of the problem

Defining the problem

Changing the representation of the problem

Developing the goals associated with solving the problem

Determining the strategy for generating the possible solutions

Choosing a strategy

Generating the evaluation criteria to be applied to solutions

Evaluating the solution criteria

Generating the solutions

Evaluating the solutions

The literature on cognitive abilities and human problem solving does confirm that individuals differ considerably, based upon their cognitive abilities (Benbasat and Taylor, 1982; Streitz, 1987), in their ability to deal with different aspects of a problem solving situation. This depends upon such psychological dimensions as their ability to deal with Abstraction - No Abstraction, Search - No Search, Data Driven - Conceptually Driven, and Deductive - Inductive cognitive processes.

In most face-to-face approaches, the group is forced as a whole to take a sequential path through a group problem solving process. In the Delphi process, we try to design a communication structure that allows any individual to choose the sequence in which to examine and contribute to the problem solving process. This is the single most important criterion by which we should evaluate the design of a Delphi oriented communication structure. Does it allow the individual to exercise personal judgement about what part of the problem to deal with at any time in the group problem solving process?

It is actually easier to accomplish this using a computer system than it has been with paper and pencil based Delphi studies. The "round" structure and the need to limit the physical size of any paper and pencil survey places severe constraints on the degree to which one can carry out the above approach. Hence, paper and pencil Delphis are usually limited by the "top-down/bottom-up" dichotomy rather than allowing more complete parallel entry to any aspect of the problem. For example, in a single Delphi one might explore on the first round "goals" (a top view) and specific "consequences" (a

bottom view). Relating goals to consequences requires developing the relationships inherent in alternative actions and states of nature. These would be put off to a later round. In the computerized environment individuals could be free to tackle any aspect of the problem according to personal preferences.

This particular objective of Delphi design is also characterized by two other practices commonly applied to Delphi studies. First, it should be clear to the respondents that they do not have to respond to every question, but can decide to take a "no judgement" view. Secondly, one usually solicits the respondent's confidence in their judgements, particularly when they are quantified judgements. This has been found to improve the quality of the estimates made in Delphi exercises (Dalkey, 1970). This allows the respondents to estimate their own degree of expertise on the judgements they are supplying. The fact that contributions can be made anonymously also means a person does not have to feel embarrassment if he or she does not feel able to confidently contribute to a specific aspect of the problem.

This advantage for the Delphi approach comes at an obvious price. With material being supplied in parallel, it is clear that the need to structure and organize it in a manner that it makes sense to the group is a primary requirement (Turoff, 1974, 1991; Hiltz and Turoff, 1985). The need to carefully define the total communication structure and put it into a framework that produces both a group view and a synchronization of the group process is the most difficult part of a good Delphi design. We will treat this in following sections. In paper and pencil Delphis, this is the effort that must be undertaken by the design team in processing the results of each round and producing a proper summary. In a computer based Delphi process, this has a somewhat different connotation in that the round structure disappears, replaced by a continuous feedback process which may or may not involve human intervention for the processing.

The most significant observation resulting from the above considerations, is that most of the attempts to understand the group problem solving process in the computer based environment are still based upon models that were developed from studying face-to-face groups. Thus, what are often thought of as being "ideal" group problem solving structures are based upon the "sequential" treatment of a problem by a group (Turoff, 1991). There has been little work to date to develop models of the group problem solving process that are based upon parallel and asynchronous activities by the individuals within the group. There is need for a model which integrates the individual

problem solving process with the group process. It is only within the context of such a model that we can come to a deeper understanding of the design process that goes beyond the trial and error evolution of the method that has occurred to date.

3.3 ANONYMITY

Perhaps the property that most characterizes the Delphi method in the mind of most people is the use of anonymity. Typically, in paper and pencil Delphis there is no identification of who contributed specific material or who made a particular evaluative judgment about it. This property is not one that should be considered a hard and fast rule for all aspects of a Delphi. Moreover, the computer makes possible variations in anonymity not possible in a paper and pencil environment. Before we explore these, we should look at the primary reasons for anonymity:

Individuals should not have to commit themselves to initial expressions of an idea that may not turn out to be suitable.

If an idea turns out to be unsuitable, no one loses face from having been the individual to introduce it.

Persons of high status are reluctant to produce questionable ideas.

Committing one's name to a concept makes it harder to reject it or change one's mind about it.

Votes are more frequently changed when the identity of a given voter is not available to the group.

The consideration of an idea or concept may be biased by who introduced it.

When ideas are introduced within a group where severe conflicts exist in either "interests" or "values," the consideration of an idea may be biased by knowing it is produced by someone with whom the individual agrees or disagrees.

The high social status of an individual contributor may influence others in the group to accept the given concept or idea.

Conversely, lower status individuals may not introduce ideas, for fear that the idea will be rejected outright.

In essence, the objective of anonymity is to allow the introduction and evaluation of ideas and concepts by removing some of the common biases normally occurring in the face-to-face group process. Sometimes the use of anonymity has been carried too far.

For example, it is important that the members of a Delphi exercise believe that they are communicating with a peer group. An individual participant must feel that the other members of the group will be able to contribute valuable insight about the problem being examined. This is a primary factor in motivating participation. It is usual to inform the participants about who is actually involved in the group of Delphi respondents. Only when there are strong antagonisms among group members would one consider not doing this.

Delphi panelists are motivated to participate actively only if they feel they will obtain value from the information they receive as a result of the process. This value received needs to be at least equal, in their minds, to the effort expended to contribute information. This is one reason why blanket invitations to participate in a Delphi that do not specify who will be involved and what the feedback will be to the group members often result in very low participation rates.

When one introduces the concept of conducting a Delphi through a Computer Mediated Communication System, there are more options available for handling the process of anonymity. First, one can easily incorporate the use of pen-names (Hiltz, Turoff, and Johnson, 1989). While this does not identify who a person is, it does allow a person to be identified with a set of related contributions. This allows the other members of a group to obtain more understanding of why specific individuals are agreeing or disagreeing with certain concepts. For example, knowing all the arguments a person has made about accepting or rejecting a given position allows people to better tailor what needs to be said to perhaps change an individual's viewpoint. It also allows the expression of more complex individual viewpoints. This coherency is hard to observe or utilize when everything is anonymous.

As a result, it is probably desirable in most computer based Delphis to impose the default use of pen-names rather than anonymity on qualitative type statements made in the discussion. In some cases it is also possible to provide the privilege of allowing the respondents to choose when they wish to use pen-names and when they wish to use their real name. The more the individuals know one another and have a history as a "social" group, the more likely that good results will result from allowing participants to freely choose to use their real names, pen-names, or anonymity on qualitative type information.

There have been studies of computer based message systems which have attempted to conclude that the use of anonymity leads to "flaming" and antagonism (Kiesler, Siegel, and McGuire, 1984). Most of these observations have been based upon studying student groups who have no prior history or knowledge about one another. Flaming and disinhibition have not been problems among groups that already have a social history or social structure. When utilizing a computer based system with groups who are not familiar with one another, it may be important to provide a separate conference devoted to socializing among the group members. This would serve the same purpose as coffee breaks serve for co-located groups that work together. In the computer based communication environment, it has been observed that social-emotional exchanges are helpful in facilitating consensus development and eliminating potential misunderstanding (Hiltz, Johnson, and Turoff, 1986).

Anonymity for voting and estimates of subjective quantitative information is probably desirable to maintain in most circumstances. However, it is desirable that the coordinator for a Delphi exercise on the computer system be able to identify people with extreme votes or estimates. A Delphi coordinator should have no vested interest in the outcome and should be in a facilitation role. The facilitator may feel it is desirable to encourage individuals with extreme positions to explain them. Sometimes the observation that one is in a minority position can negatively affect participation unless there is such encouragement.

In some cases it may be desirable to allow voter identification. For example, in the final steps of a budget allocation task, it could be felt that everyone should assume final accountability for the recommended decision. Even in face-to-face committees, committee reports where no identified individuals assume responsibility have sometimes led to a lack of group commitment when it comes to implementing the results. Also, when no one is accountable, one can sometimes get more risky recommendations than would otherwise result. This decision must be based upon the nature of the application and the group. In any case, the identification of a member's voting position should only apply to the final evaluation phase of a group process.

3.4 MODERATION AND FACILITATION

In Computer Mediated Communication Systems, aside from message systems, if one wants to conduct group oriented communications, there is still a basic need for moderation and facilitation, just as in face-to-face meetings. However, the nature of leadership in the online environment is different from that in the face to face environment.

In the online environment it is much easier to separate the role of process facilitation from that of content leadership. It is also quite easy to develop a number of different leaders for different areas of a problem.

In the paper and pencil Delphi every contribution first goes to the coordinator of the exercise and then is integrated into a single summary provided to all of the participants. Clearly, in the computer based environment, this is not necessary. Whether or not given contributions need to be screened ahead of time is a function of the application and the nature of a particular contribution. Since the individual members can update themselves on what is new before making a contribution, the amount of duplication is minimized in a computer based Delphi.

For example, it may be desirable to hold certain types of contributions until the group is at a point in the deliberations where they are ready to deal with them. Also, information such as voting results should not be provided until a sufficient number of votes about an item have been accumulated. In situations dealing with very strong controversies, it may be necessary to screen and edit the wording of certain contributions to try to minimize emotional biases and tactics such as name calling and insulting remarks.

While a lot of material in an online Delphi can be delivered directly to the group, the specific decisions on this still need to be made by the person or team in control of the Delphi process. In Computer Mediated Communications, the activity level and actions of a conference moderator can be quite critical to the success of an asynchronous conference and specific guidelines for moderators can be found in the literature (Hiltz, 1984).

There have been many Delphis where the material is summarized based upon the breakdown of the respondents into various specialized expert subgroups or differing interests and perspectives. In the computer based environment it becomes possible to consider multiple group environments. This is where there are separate communication structures or separation of the respondents into separate Delphi groups. On top of this

structure would be a higher level one that synthesizes or filters out the reduced set of information necessary to pass between the groups. This does lead to the possibility of very large populations of respondents engaged in common task objectives. A practical example of this is multiple industrial standards groups which must be informed of what is arising from other groups that impacts on their considerations, but do not need to be involved in details of the subgroup deliberations in other areas.

There are many Delphi applications where respondents actually engage in taking on roles (e.g Stakeholder Analysis, Linstone, 1984) to deal with certain situations. This requires moderator supervision and direction. Associated with role playing is the employment of gaming situations where there may be groups in competition with one another and communication is regulated by the "game director" (Hsu, 1989). In the area of policy analysis it could be very productive to allow the subgroups that have agreement about a given resolution to have a private open conference where they can discuss the best possible responses to the material in the main Delphi as a private subgroup. It is also clear that subgroups could be formulated dynamically based upon the content of what is taking place.

Multiple group Delphis in a computer environment is a relatively new potential and there are no hard and fast rules for setting up communication structures in this area. As group oriented Computer Mediated Communication Systems become more widely used, there will be much opportunity to experiment with this relatively new opportunity for structuring communications at both the inter and intra group level.

3.5 STRUCTURE

The heart of a Delphi is the structure that relates all the contributions made by the individuals in the group and which produces a group view or perspective. In a computer based Delphi, the structure is one that reflects continuous operation and contributions. This is somewhat different than the paper and pencil mode where the structure must be divided into three or more discrete rounds. As an example, we will describe potential transformations of two simple structures that have often been utilized in paper Delphis, for use in a computerized environment.

3.5.1 The Policy Delphi

The first example is the Policy Delphi (Turoff, 1970). This is an interesting Delphi structure in that its objective is not to produce a consensus, but to expose the strongest pro and con arguments about differing resolutions of a policy issue. It is a form of policy analysis that provides a decision maker the strongest arguments on each side of the issue. Usually one utilizes as respondents individuals who have the strongest opposing views.

The structure of a Policy Delphi is very simple.

Policy Delphi Structure

TYPE OF ITEM	VOTING SCALES	RELATIONSHIPS
Resolution	Desirability Feasibility	Alternatives
Argument	Importance Validity	Pro or con to a given resolution Opposing to other arguments

In the above structure any respondent in the Delphi is free to add a possible resolution (solution) to the basic policy issue, or to make a pro or con argument about one or more of the listed possible resolutions. He or she can do this at any time. Also, the respondent can vote at any time on the two types of voting scales associated with either of the item types. Individuals may also choose to change their vote on a given item at any time. In this structure the two scales are needed to highlight situations where policy resolutions might be rated in such categories as desirable but infeasible, and arguments may be

rated as important but invalid (others might believe it). When making additions of a qualitative nature, participants must also indicate how that addition is related to the existing items.

The computer's role in the above process is to organize everything so that the individual can follow what is going on and obtain a group view:

Provide each member with new items that they have not yet seen.

Tally the votes and make the vote distribution viewable when sufficient votes are accumulated.

Organize a pro list and a con list of arguments about any resolution.

Allow the individual to view lists of arguments according to the results of the different voting scales (e.g. most valid to least valid arguments).

Allow the individual to compare opposing arguments.

Provide status information on how many respondents have dealt with a given resolution or list of arguments.

The role of the Delphi Coordinator or human facilitator is very minimum in such a well defined structure. The software powers or special privileges that such an individual needs are:

Being able to freeze a given list when it is felt there are sufficient entries to halt contributions, so as to focus energies on evaluation of the items entered to that point in time.

Being able to edit entries to eliminate or minimize duplications of resolutions or arguments.

Being able to call for final voting on a given item or set of items.

Being able to modify linkages between items when appropriate.

Reviewing data on participation so as to encourage participation via private messages.

It is possible to also develop rules to allow the computer to handle some of the above functions. But in terms of today's technology, these functions are still better handled by a human. A group using this structure for the first time should go through a training exercise. The Policy Delphi structure can be designed to be fairly easy to learn and utilize. The use of graphics to support visualization of the structure of the discussion can also be helpful.

The Policy Delphi structure was first implemented in paper and pencil in 1970 and was later implemented in two separate computer versions (Turoff, 1972; Conklin and

Begeman, 1987). It should be noted that the structure of relating items in a Policy Delphi may also be viewed as a representation of a specialized or tailored Hypertext system (Conklin, 1987; Nelson, 1965). Most Delphi designs, when translated to a computer environment, do depend upon semantic relationships among items being established and are utilized for browsing and presenting content oriented groupings of the material. A generalized approach to supporting Delphi relationships within a Hypertext environment may be found in the literature (Rao & Turoff, 1991; Turoff, Rao, and Hiltz, 1991).

Most Delphi structures can be considered to be types of items (i.e. nodes) which have various relationships (i.e. links) to one another. Therefore, it is possible to view a specific Delphi as a particular instance of a Hypertext system. Hypertext is the view of text fragments in a computer as the nodes within a graph or web of relationships making up a body of knowledge. Hypertext functionality is therefore useful for the support of automated Delphi processes.

3.5.2 The Trend Model

This Delphi involves first choosing a specific trend of concern to the group. For example, this might be deaths from AIDS or the amount of life extension expected from a particular treatment. One might include in a single study a set of related trend variables. For the purpose of this explanation we will focus on one trend.

The individual respondents are asked first to make a projection of where they think the time curve will go in the next five years. Then they are asked to list the assumptions they are making and any uncertainties they have. Assumptions are things they think will occur over that time frame and which impact on determining this trend. Uncertainties are things they do not think will occur, but if they did, they would cause changes in estimates of where the trend will go.

Since some peoples' uncertainties are other's assumptions, these are compiled into a list of "possible" assumptions and every individual is asked to vote on each possible assumption according to validity. To accomplish this validity estimation the group may be provided with an anchored interval scale which varies, for example, from "definitely true" to "definitely false," with a mid-point of "maybe." The resulting list of assumptions is automatically reordered by the group validity judgement. The ones the group agrees on as valid or invalid are set aside, and the subsequent discussion focuses

on the assumptions that have an average vote of "maybe". The analysis of the voting has to point out which "maybe" votes result from true uncertainty on the part of the respondents, and which result from wide differences in beliefs between subgroups of respondents.

Clearly in the computer environment, this process of listing, voting, and discussing the assumptions can take place on a continuous basis. The voting serves to quickly eliminate from the discussion those items on which the group agrees. The remaining uncertain items usually are divided into two types: 1) those which can be influenced (e.g. improvements in knowledge about the proper use of condoms), 2) those that cannot be influenced (e.g. hospital facilities in the short term).

In the final stage, after the list has been completed and evaluated, the participants are asked to re-estimate their earlier trend estimate. One could observe that a statistical regression analysis might have produced a similar trend curve. However, the application of such a mathematical technique will not produce the qualitative model that represents the collective judgement of all the experts involved. It is that model which is important to understanding the projection and what actions can be taken to influence changes in the trend or in understanding the variation in the projection of the trend.

There is practically no planning task where the above trend analysis structure is not applicable. In the medical field, for example, considering examining trend curves for the occurrences of certain medical problems and the impact of various treatments is rather broadly applicable. This particular structure has been utilized in a significant number of corporate planning exercises. With graphic capabilities on workstations, it would be quite easy to implement in a computerized version. A similar structure may be applied to qualitative trends made up of a time series of related discrete events. An example would be AIDS cases triggering specific legal rulings and particular ethical dilemmas.

The above two examples were chosen because they are fairly simple and straightforward. However, there are literally tens of different Delphi structures that have been demonstrated in the paper and pencil environment (Linstone and Turoff, 1975) and are quite transferable to the computer based environment. Many of these require the ability to utilize graphics to view the complexity of relationships among concepts. Others require extended facilities to utilize generalized Hypertext structures. However, one of the most significant potentials for the automation of Delphi is the incorporation

of real time analysis aids for the interpretation and presentation of the subjective information produced in a Delphi exercise. This will be treated in a following section. It should also be clear from the above examples that there are certain fundamental tools that apply across a wide range of Delphi structures. The ability of a group to contribute to building a specific list, to be able to apply specific voting capabilities, and to be able to sort the list by voting results, represents a set of general tool capabilities. This is the approach we have taken in the development of the EIES 2 system (Turoff, 1991) at NJIT to support a wide variety of applications such as Group Decision Support, Delphi Design, Project Management, and Education (Hiltz, 1986, 1990).
EIES 2 is a general purpose Computer Mediated Communication System that provides many features whereby an individual moderating a specific conference can tailor the group process. The moderator of a given conference can create, at any point in the discussion, an "activity" that may be attached to a comment. These activities accomplish different specialized functions such as list collection and voting. However, the interface to all these activities is the same in the sense that the same basic generic commands apply to any activity. For example, one may "Do" the activity to make changes to it or "View" the results of the activity, regardless of what type of activity it is. The conference moderator has the authority to introduce these activities whenever he or she feels they fit within the current discussion. Also, the moderator may choose to allow or not allow the facilities of anonymity for a given activity or conference.
EIES 2 also provides a general notifications capability that can be tailored to notify the participants in a group process whenever any action occurs of which they need to be made aware. For example, a notification may let the members of a Delphi know when the votes on a specific item are sufficient to allow viewing of the resulting distribution. EIES 2 is constructed so that any programs or analysis routines developed in any language within the context of the UNIX operating system or a TCP/IP network can be integrated or made available through the EIES 2 interface. The major facility to allow a Computer Mediated Communication System to enhance Delphi processes is to provide alternative structures in the form of a collection of group support tools. The system must also include the privileges for a facilitator or group leader to decide on the dynamic incorporation of these tools in the group process.

3.6 ANALYSIS

A principal contribution to the improvement of the quality of the results in a paper and pencil Delphi study is the analysis that the design and coordination team can perform on the results of each round. This analysis has a number of specific objectives:

Improve the understanding of the participants through analysis of subjective judgements to produce a clear presentation of the range of views and considerations.

Detect hidden disagreements and judgmental biases that should be exposed for further clarification.

Detect missing information or cases of ambiguity in interpretation by different participants.

Allow the examination of very complex situations that can only be summarized by analysis procedures.

Detect patterns of information and of sub-group positions.

Detect critical items that need to be focused upon.

To accomplish the above, there are a host of analysis approaches that come from many different fields. Many of these are amenable to implementation as real time computer based support to a continuous Delphi process conducted via a Computer Mediated Conferencing System. We will briefly address here some of the most significant types of these methods for supporting Delphi applications.

3.6.1 Scaling Methods

Scaling is the science of determining measuring instruments for human judgement. Clearly, one needs to make use of appropriate scaling methods to aid in improving the accuracy of subjective estimation and voting procedures. While most of these methods were originally developed to measure human judgement, they are easily adaptable, in many cases, to providing feedback to a Delphi group on the consequences of the judgements being made by the individuals.

For example, in many cases the appropriate judgement we wish to solicit from an individual is a ranking (i.e. ordinal scale measurement) of individual items. It is comparatively more accurate to ask individuals to rank order items, such as objectives or goals, than to ask for interval or ratio measures. A person can estimate that a particular goal is more important than another one; however, how much more important it is much more difficult to estimate consistently among a group of individuals. However, a scaling method such as Thurstone's Law of Comparative Judgement

(Torgenson, 1958) can transform individual ranking judgements and produce analytically a group result which is an interval scale rather than a rank ordered scale. Providing the group the results in terms of this interval scale allows the individuals to detect in a much more reliable manner the extent to which certain objectives are clearly distinct from other objectives, and which are considered in closer proximity. Merely providing an averaging of the ranking scale does not contribute this added insight to the group as a whole. Furthermore, standard averaging approaches can lead to inconsistencies in group judgements (i.e. Arrow's Paradox). This can occur when there are disagreements underlying the averaging and when there is a lack of appropriate "anchoring" of the scales.

Standard correlation analysis approaches can be utilized to determine if there are subgroups or patterns of agreement and disagreement that exist across different issues or judgements made in the Delphi exercise. Do the people who feel a certain way about an issue feel the same way about another issue? This type of analysis should, in most cases, be provided first to the facilitator, and that person should decide which relationships need to be passed back to the group. In many Delphis, there are identified sub-groups. A Delphi might comprise people from different disciplines. Do the administrators, researchers, lawyers, insurers, and practitioners have differences in viewpoint that are based upon the perspective they take on a new medical treatment? The utility of these insights needs to be evaluated by the facilitator in the context of the application. With groups that work together over a long term, it might be desirable to provide such an analysis in terms of direct feedback without facilitator intervention.

Scaling methods span a wide range of techniques, from fairly simple and straightforward to fairly sophisticated. An example of a sophisticated approach is Multi-Dimensional Scaling (Carroll and Wish, 1975). MDS allows subjective estimates of similarity between any two objects to be translated into a relative position in a Euclidean space. It provides, in essence, N dimensional interval scaling of similarity estimates. The number of meaningful dimensions found suggests the number of independent dimensional factors underlying the way both the individuals and the group are viewing the similarity among objects. By looking at the alternative two dimensional projections, it is possible to arrive at an understanding of what the dimensional factors are.

The process by which one would use MDS in a Delphi would be to ask for the similarities and provide back the graphical layouts of the alternative dimensions. The respondents would then be asked to try to determine what these dimensions mean or represent. The result is a very powerful technique for potentially exposing the hidden factors a group is using to make judgements about similarities. The question of similarity is one that can be applied to a very wide range of object types, e.g. goals, products, countries, relationships, jobs, criteria, etc. MDS may also be viewed as a form of Cluster Analysis, and many methods in Cluster Analysis (Anderberg, 1973) can also be usefully applied to analyzing the subjective comparison judgements made by Delphi respondents.

When a group is using voting and estimation structures over a long period so that they make judgements about a growing number of similar situations, it is possible to consider the introduction of "scoring" methods (Dalkey, 1977), into the Delphi process. Given later feedback upon the accuracy of estimates or the quality or success of a given judgement, it is possible to consider feedback to estimators on their degrees of "accuracy" or possible biases due to factors such as conservatism. At the point where there are individuals utilizing Delphi techniques on a continuous basis, it will be possible to conduct the sorts of investigations needed to develop this particular area as a decision aid.

Designing a Delphi, whether via paper and pencil or on the computer, does include the process of designing a survey. As such, all the guidelines on good survey design and all the analysis methods that have been developed for analyzing survey data are potentially applicable to a Delphi. There is, however, a fundamental difference in objectives, which determines how one employs a given method, and whether it is applicable in a given situation.

Most scaling methods were evolved to aid in making an assessment of a human judgement with the premise that one is measuring a stable and constant quantity. One's intelligence or personality would not be affected or changed as a part of the measurement process. The goal is to discover biases and inconsistencies and to produce more accurate measurements. In the Delphi process, however, we are interested in informing the respondents about what they are really saying, and how it compares to the group as a whole. We are also interested in promoting changes in viewpoints and the other items we measure, if it will promote reaching a superior group view of the

situation. We are also interested in detecting and exposing hidden factors or relationships of which the group may not be completely aware. With this in mind, one has to take special care that the use of these analysis methods does not convey a false impression of finalization in a group view.
Related to scaling is the area of Social Choice Theory, which provides alternative methods for the summarization of voting processes (Hogarth, 1977). The use of multiple methods of viewing the summarization of a given voting process can be useful in preventing a group from placing an over emphasis on a single voting result.
Probably the most important single consideration in the past that has prevented the incorporation of many of the approaches discussed here is the difficulty of educating the respondent in the interpretation of the method when the respondent is involved in only one short term Delphi process. With the potential that Computer Mediated Communications offers for long term continuous use by groups, it is now possible to consider incremental training for individuals to gain an understanding of the more sophisticated methods.
With the appropriate use of scaling methods it becomes possible to establish that individuals will mean the same thing when they use terms like: desirable, very desirable, likely, unlikely, agree, strongly agree, etc. It becomes possible to determine which alternatives are truly similar and which are distinctly different. Scaling methods, in essence, serve the objective of eliminating ambiguity in the judgmental and estimation process of a group.

3.6.2 Structural Modeling

The term Structural Modeling (Lendaris, 1980; Geoffrion, 1987) has come to represent a host of specific methods that have the objective of allowing an individual to express a large set of independent relationships and judgements which the given method utilizes to produce a "whole" model of the "system" being described. In computer terms, these are methods that allow a user to build a model of a situation without having to program or go through the use of experts in modeling and simulation. These methods vary from ones that provide a simple static relationship model (e.g. Interpretive Structural Modeling, Warfield, 1974), to more dynamic probabilistic and time varying models

e.g. Cross Impact, Time Series Regression, etc.). Just about any technique that organizes data into some sort of framework is a candidate for falling under the rubric of Structural Modeling. This includes Decision Trees and Payoff Matrices.

The objective of these approaches is to allow participants as individuals or as part of a group, to contribute pieces of a complex situation and to be provided a composite model. For example, in Interpretive Structural Modeling the individual is asked only to make a series of judgements about each two components of a model (such as two goals) with respect to whether they are related. The resulting complex network of relations is analyzed to collapse the network to a hierarchy of levels utilizing the existence of cycles within the network to make that simplification. The result for the individual or group is a set of levels or clusters of the objects which infer a relationship of higher to lower levels. This provides a graphical representation of the binary judgements made about each set of objects, taken two at a time.

The Cross Impact type model allows individuals to express probabilities of occurrence for a series of events, and conditional probabilities based upon assumptions as to which events will or will not occur. This is used to construct a quasi-causal model that allows participants to then vary the original estimates of individual events and see the consequences on the whole event set.

An excellent example of structural modeling to determine the important relationships and impacts on changes in medical care policies may be found in a recent article by Vennix et. al. (1990). This particular example is based upon the specification of negative and positive feedback loops. The development of the model was arrived at through the joint use of a paper and pencil Delphi and follow up face-to-face meetings. All these techniques may be used in a Delphi process to help a group to develop a collaborative model of a complex situation. This is one area where the merger of the Delphi process and the computer presents a unique opportunity for dealing with situations of unusual complexity. More often than not, the individual experts who can contribute to building a complex model are geographically dispersed, and the effort to derive and improve such models is one that needs to take place over an extended period of time. In other words, improvement of the model has to be based upon feedback from its performance and incremental refinement.

A recent experiment (Hopkins, 1987) produced the very significant finding that it was possible to distinguish the degree of expertise an individual had about a complex

situation by the measured richness of the models that were specified by each individual. This finding suggests the possibility of incorporating automated procedures for rating potential quality or inferred confidence in the contributions made by various individuals. This possibility deserves further investigation, as it would obviously provide improved models with a reduced communication load requirement.
Developing all the structural relationships in models of symptoms, tests, diagnosis, and treatment is an obvious area for the application of structural modeling. Appropriate techniques can be utilized on the computer to allow individuals to visualize the structures resulting from their contributed relationships and examine that structure for consistency. At the group level the same methods can be used to examine composite models for consistency and feed back inconsistencies for further refinement. Individuals are good at estimating individual relationships, but they are not always able to maintain consistency in developing complex models. The problem is compounded for group efforts.
A group can improve the nature of a model only by first seeing the results and consequences of the current design. Model building is a long term incremental process. The proper integration of Delphi methods, Computer Mediated Communications and Structural Modeling methods makes possible effective large scale modeling efforts not otherwise currently doable.

3.7 DELPHI, EXPERT SYSTEMS, GDSS, AND COLLABORATIVE SYSTEMS

The concept of an Expert System is to somehow capture the knowledge of a group of experts and store it in a computer for utilization by non-experts. The incorporation of the Delphi method in computer environments makes possible a number of significant refinements of this objective and some fundamental possible changes to the nature of Expert Systems.
The common approach to the development of an expert system is to achieve agreement among all involved experts before the actual coding of the knowledge base is performed. At present this is accomplished by a knowledge engineer or team of knowledge engineers, who must interface to a team of domain experts. Besides being time consuming, the fundamental flaw in this approach is that even within scientific and/or engineering fields, there is incomplete agreement among experts. Furthermore,

agreement and disagreement are evolving properties that change dynamically over time. The Delphi method may be viewed as an alternative approach to collecting and synthesizing expert knowledge. In fact, within the current terminology, the design of a Delphi is in fact the design of a knowledge base or structure for putting the collected information together. It has also been an important objective of Delphi design to capture disagreements as well as agreements.

Another potential problem area is that experts concerned with a common problem can be in conflict. For example, design, production, and marketing professionals can have severe conflicts about the properties of a potential new product. Different medical researchers have different views about the most promising directions for research. Some of the problems addressed here have been investigated in the work on "Multi Expert Knowledge Systems (MKS) (LeClair, 1985, 1989). LeClair's work represents one of the few in depth approaches to incorporating the knowledge of disagreeing experts into the same system. However, this work still assumes the final system no longer incorporates the humans, but only their knowledge.

On the other hand, the view that we believe is the most promising is an objective for "Collaborative Expert Systems," where the experts are provided a knowledge structure (a Delphi Design) that allows them to dynamically contribute their knowledge to the system and to modify and evolve the system, over time. Clearly, such a system is one which the experts must desire to use for themselves as well as a tool for others who need their knowledge. This is the situation where the experts are both the creators and the users of the resulting expert system.

Without the above form of expert systems, the only feasible systems are those that restrict themselves to well established rules and agreements. In our view, the future of expert systems lies in their ultimate ability to be utilized by working groups of experts as a tool for collecting and assessing their collective knowledge about their work.

The current approach to expert systems through the use of knowledge engineers has been recognized as the chief bottleneck to the creation of these systems (Welbank, 1983; Waterman, 1986) for four main reasons:

Human expertise is usually complex, undocumented and consists of many different types and levels of knowledge (e.g. casual knowledge, common sense, meta knowledge, etc.)

Different experts may solve the problem differently and therefore may argue or even criticize one another on the method used.

There often exists a communication barrier between the knowledge engineer and the experts. The knowledge engineer is not an expert on the area and many experts do not understand their own problem solving process. As a result, many details and complications of the reasoning process may be ignored or obscured.

Motivation for the expert is often lacking because the results are often delayed or are not intended to benefit the expert.

Many of these problems can be overcome if one can develop collaborative design systems that focus on allowing a group of experts to develop their own expert system in an evolutionary manner and as a group oriented aid to their own work. The evolving system could also be tapped by non-experts for use. In that mode it would be considered by the experts as an aid to disseminating needed information to a wider circle of users and freeing the time of the experts for more difficult problems.

A collaborative expert system has to deal with at least four types of knowledge:

Deductive reasoning as represented by rule based models.

Inductive and intuitive reasoning representing experience on the part of experts.

Objectives, Goals, and Vested Interests which are viewpoints of experts in given circumstances.

Values and Beliefs which often underlie judgements about viewpoints.

The first two types have been typical of current expert systems. The other two areas have largely been the domain of Group Decision Support Systems, Delphi and Nominal Group Techniques, decision and utility theory, and psychological measurement methods. All four of these types of knowledge in a collaborative expert system must handle disagreement among the participants.

3.7.1 The Deductive Level of Disagreements

At the predicate logic level, experts may disagree about both the predicates to use and the rules that are valid in the real world. A well designed knowledge acquisition and expert environment should permit experts to "speak their mind" and not limit them to a preconceived vocabulary. It is therefore necessary that the accumulation of the vocabulary for specification of the rules be an integral part of the collaborative process. Even if experts agree on a basic vocabulary, they often disagree about subtle details of a representation. This problem occurs whenever there are several possible reference frames, a situation which is well documented in the literature (e.g. Sondheimer, 1976). Unfortunately, at the current state of the art, two relations with different numbers of arguments are treated by logic programming environments as being two completely different entities.

One approach to this problem is to allow each member of a collaborative group to construct and tailor their own knowledge base and then to superimpose an analysis system for determining various types of agreement and disagreement. There are various weighted voting procedures (Shapley and Grofman, 1985) and scaling methods (Torgerson, 1958) that are promising for analysis of this situation. Weights have been used in some expert systems (e.g. Reboh, 1983). When such information is being accumulated over time then there are various "scoring" approaches (Dalkey, 1977) that may also be employed and coupled with "explanation based learning" approaches (Pazzani, 1988). Early work with the Delphi method indicated that even experts in a given area differ in expertise in various sub-domains and that the greatest improvement in accuracy of estimates was obtained by weighing estimates by this type of difference (Dalkey, 1970).

3.7.2 The Inductive Level of Disagreements

One of the major problems in designing knowledge representations that reflect common sense models of the world is that the world is not a discrete and well specified place. In fact, the world is quite vague and ambiguous. Ambiguity is the key property that most people have to deal with in reaching conclusions and decisions (Daft and Lengel, 1986). Ambiguity results from differences in concepts (e.g. "expensive") among different people and, in this context, from the collaborative process itself. In many cases the problem of ambiguity can be structured as the degree to which an object "more or less belongs" to a class. Fuzzy sets (Zadeh, 1965; Klir, 1988) are a generalization of standard sets that allows for degrees of membership. One approach to this problem is to

utilize fuzzy set theory to represent the types of ambiguity that result from intuitive thinking.
The major research issue in this area is to develop methods for accurately combining multiple judgements and resolving disagreements about estimates of degrees of membership in fuzzy set relations (Stephanou and Sage, 1987). In this instance, scaling methods seem particularly appropriate. Humans are good at ranking (object A belongs more than object B) but not good at direct estimates of correlation factors needed for fuzzy set relationships. However, various scaling methods can be used to convert a collaborative set of ranking measures to interval or ratio scales.
Another approach is to incorporate multivalent and fuzzy logic (Dubois and Prade, 1980) into any model framework where the expert group is building the relationships. An example of degrees of truth and the resulting treatment of logical inference from a fuzzy perspective may be found in Baldwin (1981).
In essence, the problem is the recognition that models that capture intuition have to capture the structure of disagreements. A result is no longer true or false, but possibly a little of both. Rather than a group process being dedicated to eliminating disagreement, the objective is to capture it, quantify it, and integrate it into the collective model. There has always been a bias that disagreement has no place in the result of a scientific process. Because of this we can become blind to the forcing of unwarranted consensus. It would be a far more realistic view of the world to recognize the necessity for disagreements and "fuzzy" relationships as a fundamental part of any model meant to reflect the collective intuitions of a group of experts.

3.7.3 Goal and Value Disagreements

This is the area that is typically included in applications of Group Decision Support Systems. While there are certain specific approaches (e.g. Stakeholder analysis) for eliciting this type of information, the current state of the art is largely the use of human facilitators to guide the group process for the treatment of this type of knowledge. The fundamental issue of how far one can go in the process of substituting computer facilitation for human facilitation is very much an open issue. Earlier experiments in this area (Turoff & Hiltz, 1982; Hiltz et. al., 1986, 1987) showed that under some circumstances computer facilitation can degrade the performance of the group.

The approach that seems to be the most promising is to evolve a collaborative expert system that would be used to guide the meta group process. This would suggest to the group at what points in the activity they should shift the nature of what they are doing. However, such a facility would also be tailorable by the group so that it can gradually adapt to the preferred group process. Such a system would have to employ "default reasoning" approaches (Post, 1990).

As can be seen, there is no fixed dividing line between such areas as Delphi, Computer Mediated Communications, Group Decision Support Systems, and now Expert Systems. The concept of "collaborative expert systems" is really based upon the foundations established in each of these other areas. Subjective estimation, collaborative judgement formulation, and voting are strongly related support areas that also contribute to the potential for design in this area.

3.8 CONCLUSION

Delphi, as a tool, has reached a stage of maturity in that it is used fairly extensively in organizational settings in either the paper and pencil mode or in combination with face-to-face meetings and Nominal Group Techniques. Since most of these exercises are proprietary in nature there is not much of this activity reported in the open literature. The one exception to this is the applications in the medical field which are in fact actively reported and documented (Fink, Kosecoff, Chassin, and Brook, 1984). This clearly is a result of the growing need to formulate collaborative judgements about complex issues that are associated with the production of guidelines on medical practice and decisions.

Computer Mediated Communications has also seen some very significant applications in the medical field with respect to the formulation of collaborative judgements. One of the most significant to be reported in the literature was the use of leading researchers in Viral Hepatitis to review the research literature and update guidelines for practitioners (Siegel, 1980). While this was not run in an anonymous mode, it had all the other aspects of structure necessary for a dozen experts to deal with some five thousand documents and reach complete consensus on the resulting guidelines.

Another CMC application that had Delphi like structuring with Anonymity was a Group Therapy process to aid individuals in the cessation of smoking (Schneider, 1986;

Schneider and Tooley, 1986). A general review of CMC applications in the medical field can be found in Lerch (1988).
However, there is yet to be a true merger of Delphi with Computer Mediated Communications. It is only now that the technology is becoming generally available to support the high degree of tailoring necessary to dynamically structure communications within a single conferencing system (Turoff, 1991). Most conference systems, to date, have only represented single design structures with very little control available to facilitators and moderators of discussions. Also, the general lack of graphics has placed a considerable limitation on just what Delphi techniques could be adapted to the computer environment. The merger of Delphi and Computer Mediated Communications potentially offers far more than the sum of the two methods.
Long before the concept of Expert Systems it was known that statistical factor models (Dalkey, 1977) applied to a large sample of expert judgements could produce performance that was consistently in the upper quarter of the performance distribution curve. Such models did not suffer from "regression to the mean" and could result in matching the best decisions by the best experts in the group. Expert Systems is really the emergence of tools to allow this to be done on a fairly wide scale. However, the results of Expert System approaches, as currently practiced, are never going to do better than the best experts.
The merger of the Delphi Method, Computer Mediated Communications and the tools that we have discussed opens the possibility for performance of human groups that exceeds the composite performance curve. We have termed this phenomenon "collective intelligence" (Hiltz and Turoff, 1978). This is the ability of a group to produce a result that is of better quality than any single individual in the group could achieve acting alone. This rarely occurs in face-to-face groups.
A recent experiment in utilizing human judgement in conjunction with the types of models that are used in Expert Systems confirms that this is in fact possible (Blattberg and Hoch, 1990). There has been too much attention in recent years to utilizing computer technology to replace humans and far too little effort devoted to the potential for directly improving the performance of human groups. This can be achieved through integration of computer based methods and the concept of structured communications at the heart of the Delphi Method.

CHAPTER 4

ARCHIVE SYSTEM

An introduction to principles of organization and description used in archives
A first step towards acquiring the knowledge and skills needed to provide access to the rich cultural heritage information in archival collections.
Links to additional resources for further archival training such as workshops, readings, professional organizations, archival education programs and conferences.

4.1 Basic concepts and principles of archives and records management

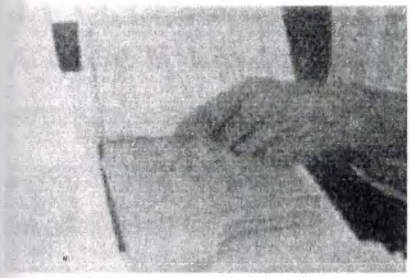
Through the centuries, the following THREE FACTORS have shaped the concepts, principles and techniques which records/archives managers use to carry out their responsibilities:

4.1.1 The Inter-relatedness of records

Because records are the documentary by-products of work or life processes, they are, like individual frames of motion picture film, organic bodies of related material which cannot be used in isolation or separation from one another without loss of integrity and meaning. They are unselfconscious in that they are naturally occurring contemporaneous and candid documents, as opposed to individual documents created intentionally for the purpose of 'history'.

4.1.2 The central importance of context

Records draw their significance from their context. That is, they are valued or useful only in groups and only in relation to the activities and purposes for which they were created and used. Thus records/archives managers must accurately identify and explain both the context of origin/creation and the context of use/custody and maintain the records in a way that preserves their original character and relationships as 'bounded entities comprising content, structure and context'.



Records rescued from past neglect often come to the archives in a disordered mess. Archivists must spend time carefully examining material to identify and restore its original provenance and order before it can be used by researchers.

4.1.3 The function of records as evidence

As we have seen, records represent or stand for human experiences, transactions, activities or accomplishments. The records designated as the 'official' or 'record copies' of documents have been selected to endure as unique testaments, all other duplicates, whether exact copies and different formats having been destroyed. They provide objective 'proof' that something has happened or been agreed to by consenting parties and as such have an integrity that must be protected and preserved by responsible and continuous custody and properly authenticated if that chain of responsible management is questioned.

Essential characteristics of recordness

What distinguishes records from other information entities is summed up in the term 'recordness'- an elusive quality usually represented by the six characteristics described below.

4.1.3.1 Recordness requires that records are:

4.1.3.1.1 Complete

A record is considered complete when it is a finished, bounded entity comprising structure, content and context and when it has the following elements: date(time and place of creation, transmission and/or receipt); originating address, an author/compiler, an addressee/recipient, title or subject accompanying its content/message.

4.1.3.1.2 'Fixed'

Records are information presented in a static form. The act of recording 'freezes' the relationships among the information elements and embeds them in a structure, which can be replicated or re-constituted. For example: data elements within a dynamic or 'live' database are not records until a transaction incorporates such data elements (content) into a meaningful and re-creatable format (structure) along with essential specific contextual information (context). The resulting record is a 'bounded entity' - a metadata encapsulated object (MEO) which may be retrieved and re-presented for human use in electronic form (screen display) or hard-copy printout (paper or film). Although a record is a 'fixed' object, its role is dynamic, not static. Subsequent records and business activities change its relationships, significance and meaning as time passes.

4.1.3.1.3 Organic

Records are the natural output of work processed; therefore records are only meaningful as a sequence of transactions. Each record is related to or is a consequence of some preceding document, with the matters documented by the former further explained or dealt with in the latter.

4.1.3.1.4 Contextual

Records derive their meaning, and therefore their usefulness or value, from their context. Because they are created, organised and used in the conduct of normal business by a particular entity, they reflect the purposes and the activities of that entity over time.

Thus, to use/appreciate the records one must understand their function, control system, relationships and pattern of use in the workplace; correspondingly the records comprise the evidence of the policies, activities and transactions of that creating entity.
For this reason, records are managed not as individual items, but in aggregates known as series, which are organic 'flows' of material created and structured by the normal course of work activity.

4.1.3.1.5 Authoritative/'Official'

Records created as documentation to support ongoing work or business activity have a status as 'official' evidence of those decisions and actions undertaken 'in the normal course of business', whether that 'business' is commerce or art or literature or just living. That is, they embody the full and unchallengeable authority of the author - be it an individual or an organisation and its officers.

4.1.3.1.6 Unique

Unlike books, journals and other published material, records appropriately maintained in context are always one-of-a-kind. That is, any record or group of records with its sequences and interrelationships is unique, despite the fact that there may be duplicate copies of some or even all individual records held elsewhere. The point is that each copy will have a different relationship to the conduct of business and to those who authorised or participated in it. However, in cases where duplication is extensive, it is usual for recordkeepers to minimise the number of duplicate series of records in their care by designating the most authoritative set as the 'official record' for retention and directing those of lesser importance to be destroyed.
Foundation principles for organising and keeping archives
With these three underlying factors and related characteristics providing the intellectual background, let us now explore the Foundation Principles of modern archives management.

4.1.3.1.6.1. The principle of provenance

This fundamental principle of grouping requires that records be organised and maintained according to their transactional origin or source. In short, records originating from one office or individual form a distinct body of material, which is to be

Records must be managed and accessible within an archive so that their provenance, original order, and chain of responsible custody is established and maintained
kept separate and inviolate. It must not be intermingled with records of other 'parentage'. Records from the same origin came to be known as 'fonds' and the principle as Respect des fonds, reflecting its French popularisation and as provenienzprinzip in German. The thinking behind this principle reasons that for records to serve as evidence, they must be traceable to their source and be shown to reflect their contexts of origin/creation and initial or primary use.

Practically speaking, this principle was adopted to preserve the chain of accountability within the ever-growing number of records documenting similar functions and activities produced by growing bureaucracies. It was important to know who was initially responsible in each transaction and to maintain an authoritative custodial lineage

4.1.3.1.6.2. The principle of original order

The Prussian (later German) archivists of the mid-19th century expanded the influence of the office of origin by developing and establishing the related Principle of Original Order or Registratorprinzip in German. This maxim stipulates that records are to be maintained in records/archives repositories in the same scheme of order and with the same designations they received in the course of the business of their office of origin and primary use. Again, the emphasis was on establishing the authenticity and integrity of the record as evidence of work processes and activities in context.

4.1.3.1.6.3. The chain of responsible custody

Completing the trilogy of context and process oriented principles for records/archives management, we come to the Principle of Continuous Custody. Articulated and popularised by the English archivist, Sir Hilary Jenkinson, this principle focuses upon the role of records/archives as evidence and maintains that evidential integrity can only be ensured when we can trace 'an unblemished line of responsible custodians'. By responsible, Jenkinson means committed to the 'physical and moral defence' of the archives. These phrases embody the responsible manager's obligation to ensure both the physical security and the intellectual integrity of the records as evidence. This faultless lineage is, in Jenkinson's view, a reasonable guarantee that the records have been kept without damage, alienation, improper or unauthorised alteration or destruction.

Protection of the essential role of records as evidence is the wellspring from which all concepts, theories, principles, policies and practices of sound recordkeeping arise. Any new approach or technique for managing records must adhere to them or acknowledge their centrality and fully justify any modification or proposed departure from them.

4.1.3.1.6.3.1 What goes on in the archives?

Because of the power and importance of records as a resource, managing records over time is a necessity. Management means imposing a regime which influences the control, accessibility, disposal, and storage of this irreplacable evidence and which manages itself effectively.

Regimes adhering to these core functions ensure that reliable records of the highest quality and integrity are available in a timely fashion for authorised use at the right price. These processes further guarantee that the best of the records continue to be available effectively and efficiently as part of our cultural knowledge base.

4.1.3.1.6.3.2 CADSS

The functions embodied in the acronym CADSS (for Control, Accessibility, Disposal, Storage and Sustain) make up the core of all information management activities.

4.1.3.1.6.3.3 CONTROL

The CADSS management model begins with C symbolising the function of Control. Whilst the word 'Control' also conveys the overall goal of records/archives management, Control as a discrete function includes all recordkeeping activities required to:

bring a record into existence as complete, integrated entity that can serve as reliable evidence of the acts to which it attests;

identify the physical and intellectual attributes of a record's content, structure and context;

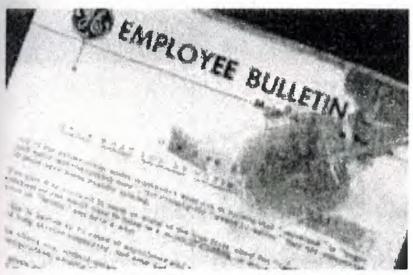
devise/identify and assign a physical 'address' for the record so that the record can be safely stored and efficiently retrieved;

articulate/represent and document the attributes and 'address' above as an integral part of the record and of the information systems controlling access and retrieval of the record and/or of information about the record

Machine-readable records such as video and audio tapes are more complex to describe than paper-based materials because understanding and accessing them needs special machinary. Documenting how these records were made and what technologies were used in their making is an integral part of the 'Control' function of record keepers.

In this photo, an archivist makes notes about the content of an oral history tape, relating it to numbers as registered by a counter on the tape recorder. Future listeners should be able to find particular content without listening to the whole oral history.

In creating/custodial offices, Control would include all tasks associated with creating and/or receiving and organising records physically and intellectually for initial use. If there is a centralised or coordinating entity such as central registry, records management and/or information services with this responsibility, there is a chance for cohesion and continuity.
However, the likelihood that these entities have been discarded or ignored in the technologically driven decentralisation of management and concurrent elimination of clerical support (more chiefs, less indians) means that there may be no records system-wide coordination over such critical factors as file content, structure, titling or indexing/access points, nor would any given record necessarily have an inviolable, integral link with its context of creation.
You never know what you might encounter in a body of records. Ham sandwiches, cockroaches, peanuts, well chewed gum, even mummified birds have been uncovered when arranging and describing abandoned masses of files.
However in this case, the lollypop is not detritus, but an integral part of the record. This special circular was sent to employees by a trade union seeking members support against the company.



In traditional archival repositories, Control embraces a range of activities known as accessioning and arrangement and description. These processes document the nature and origins of the material and explain them to prospective researchers.
In many cases archivists must conduct extensive research to recover or re-construct information needed to explain records that was lost long ago. In this work, they may employ investigative principles and techniques similar to those used by archaeologists.

4.1.3.1.6.3.4 ACCESSIBILITY

The function of Accessibility covers all activities associated with determining, administering and facilitating access to and use of records/archives. Accessibility may involve the acquisition or design and operation of specialist facilities, services, expertise and information sources to ensure that:

laws, regulations, conditions and terms of access to records/archives are suitable, authoritative, documented, disseminated and properly administered;

information about the records/archives, about obtaining access to them and about using them effectively is accurate, understandable, timely and readily available to authorised users;

prospective users and their uses are appropriate, authorised and documented;

the records/archives are retrieved, used and returned to safe custody in a timely manner.

Providing access to images can be difficult as pictures do not translate well into words and producing full size photographs of images is costly. One solution which achieves access and preservation objectives is to copy the most important images, creating a security negative for preservation and a miniature contact print to include with the finding aid in bound form or in a loose card file as shown here.

With the advent of computers and scanning technologies, many of these manual files can be scanned and made accessible electronically. Image databases can provide digital access to important photo collections either on site or over the Internet

In creating/custodial offices, Accessibility is represented by the systems and practices that govern which staff may see and use records and the mechanisms for controlling and documenting authorised access and use. Common activities include checking passwords and clearances,monitoring record movements documentation and auditing use.

In traditional archives, the activities of Accessibility are called 'reference services' and centre around a designated reading or research room facility.
Accessibility may also involve the creation of more detailed or subject oriented finding aids such as indexes or special guides or lists. These tools create additional points of access (by subject, name of participant, date, type of record, geographic location) which complement the basic structural finding aids based upon provenance.
Client-oriented training and education to promote the use of archives or to develop research skills is also included in the accessibility function

Providing a suitable environment for research work is a continuing challenge. This supervised reference facility is spacious, quiet and well furnished for comfort and efficiency.
In the public sector, both archives repositories and creating/custodial offices also have a general responsibility to make information and/or records available to the public under such legislation as Freedom of Information, Privacy and Archives or Public Records Acts.
Popular finding aids, such as this one , are paying a high price from overuse. Part of reference management is to prevent such occurences.

4.1.3.1.7 DISPOSAL

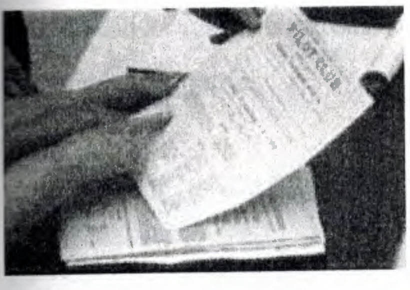
The Disposal function incorporates all activities involved in:

identifying and documenting/surveying organised activities and records making/keeping systems within a designated universe of records keeping responsibility;

determining what documentation reflecting organised activity within a designated record making/keeping system should be retained and, conversely, what should be destroyed;

authorising when, where, how and by whom these decisions will be implemented and documented;

providing advice, mechanisms, facilities and documentation for systematic, secure and accountable disposal processes and outcomes.



Properly designed disposal instructions inform employees of how and when to close files, of where and how long to store them, and ultimately, whether they are to be destroyed or preserved indefinitely as archives.

In creating/custodial offices, Disposal embraces activities such as:

conducting records use/location/activity audits;

providing information about records use/location/activity for disposal decision-making;

drafting or responding to proposed disposal decisions and recommendations;

approving, implementing and reporting disposal actions.

In traditional archives, the Disposal function is generally manifest in activities to identify and select or appraise/evaluate records of enduring value, but may also include the disposal and de-accessioning of unwanted material following appraisal or subsequent re-appraisal.

4.1.3.1.8 STORAGE

The Storage function is largely concerned with the physical preservation and care of records and archives through the use of archivally sound and/or appropriate :

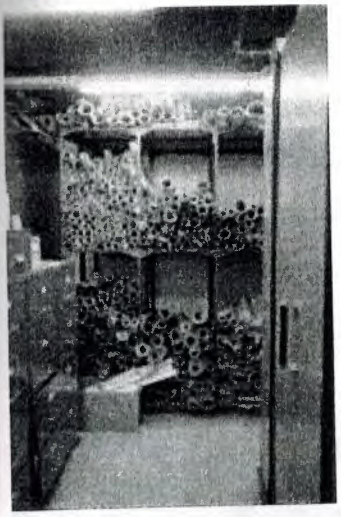
recording technology and media, packaging components/supplies, storage equipment, facilities, macro- and micro-environments;

handling procedures during retrieval/refile, use, copying, display, transfer and transport;

macro-preservation actions including:

- risk assessment and minimisation;
- preventative and protective intervention activities;
- disaster response and recovery planning;
- collection and environmental stability monitoring; and
- informational copying/media migration;

micro-conservation treatments to stabilise, repair, strengthen and/or protect individual documents or series



Training programs to educate record creators and support staff in the proper care of long-term records can help eliminate destructive practices such as those evident in this nightmare storage room.

What threats to record preservation and integrity can you see in this picture?

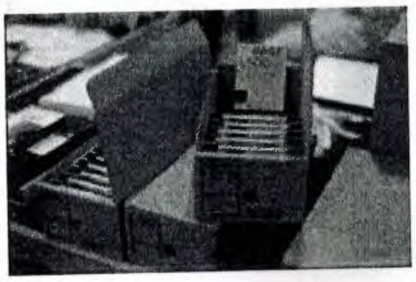
4.1.3.1.9 Check Answer

Once cleaned and de-acidified, single, oversized items such as plans, posters, prints, and drawings are ideally stored inside clear polyester 'envelopes', easily constructed from conservator-approved materials as shown in this photo. Known as encapsulation, this procedure protects items from dust and handling

Within creating/custodial offices, Storage can be badly fragmented. Decisions affecting the choice of record making technology, media, components/supplies and equipment are frequently divided among the Information Systems or IT services section, purchasing, central records/registry (if it exists) and individual office managers of decentralised records systems.

Standards and procedures for records and file maintenance and handling can be chaotic if no central policy or coordinating responsibility exists.

In traditional archival repositories, the Storage function may be shared between two organisational entities- program administration which frequently manages the plant/facilities and technical and/or preservation services, which may also include centralised microfilming, photographic and electrostatic copying.



Specialist forms of records require appropriate packaging to facilitate access and ensure preservation. Here, audio tapes are boxed in acid free cartons to protect them from dust

4.1.3.1.10 SUSTAIN

The final CADSS letter represents Sustain, which is used here as a synonym for management - the function that sustains the recordkeeping regime as a viable and effective component of its host organisation.

Whether a particular operation is large or small, all professional recordkeepers must fullfil their management responsibilities to acquire and deploy valuable resources and to get work done productively, effectively, harmoniously. Know what ensuring essential evidence through effective recordkeeping requires and how it can be achieved in various contexts.

Set realistic written objectives which complement and support the overall purpose and strategies of your host organisation.

Identify others who require essential evidence in their work and involve them meaningfully and appropriately in the process of setting and achieving these objectives.

Acquire and deploy resources (staff, facilities, funds) to meet these objectives.

Use the agreed upon objectives and distributed work tasks as the focus to encourage cooperation/collaboration and achieve productivity (management by objectives).

Assess the quality and quantity of progress and end results (evaluate performance).

Obtain and disseminate essential knowledge, skills, techniques, attitudes on a continuing basis.

Recordkeeping staff are essential to the smooth running of an organisation and require appropriate training and resources to effectively fulfill their roles.

Because recordkeeping regimes must establish and operate effectively across the whole organisation, it is essential that recordkeeping professionals understand and learn to enhance the influence and, thereby, the effectiveness of their recordkeeping regimes. The pathway to success is smoother when one or more of the following conditions exists or can be created:

A workplace culture that fosters productive, professional relationships and welcomes diversity.

A constituency (or at least those members which are themselves powerful or close to power) which knows/ understands/ appreciates the function of recordkeeping.

A host body which itself employs the regime for a full ranges of recordkeeping services, including protection of its own records with enduring value or archives.

An integrated archives/records management program which provides services to meet the host business, regulatory and cultural/historical recordkeeping needs.

An administrative placement and structures that facilitate ready access to key decisionmakers; are linked to powerful units with authority across organisation and/or are not too far down the chain of hierarchy.

A host body with a cohesive focus on recordkeeping, not just on administering a 'heritage' or 'culture'. For example, archival collections may be administered as cultural heritage, along with a museum, art gallery and/or library materials. When archives are administered strictly as cultural objects split off from the recordkeeping regimes that generated them, those powerful organic relationships that link them to ongoing management effectiveness and regulatory accountability are broken. In such cases, archives may appear restrictive and mundane in comparison with more accessible, visual, understandable and, therefore, sexier, more exploitable art and museum material.

Visibility and networking are crucial to recordkeeping effectiveness. As the sign attests, the archivist of this college has positioned the archive for optimum contact with the powerful.

4.1.3.1.11 Recordkeeping regimes

The professionally managed recordkeeping regime = A rewarding investment

Global diversity and complexity is placing greater and greater emphasis upon recordkeeping systems. Good recordkeeping doesn't happen automatically. The design and operation of recordkeeping regimes ie. the programs for making and managing records, requires specialist professional knowledge and skill. And, as with all worthwhile enterprises, you must invest appropriate resources to achieve effective results.

4.1.3.1.12 Where we work: The recordkeeping context

As stated earlier, everyone needs and keeps records, though some do it more extensively and formally than others. Generally, recordkeeping specialists do their work within public or private sector organisations as part of work units bearing some variation of the title: Archives and Records Management Services.

4.1.3.1.12.1 Public sector placements

A public recordkeeping authority that oversees the capture and maintenance of evidence on behalf of 'the people' performs a duty of care that requires objectivity. Thus it may be established as an independent body, or, provided regulations protecting its integrity are present, it may be part of a larger agency administering centralised management or heritage responsibilities.

Independent Body - Statutory authority or corporate entity, often with an advisory board composed of public 'watchdogs', industry experts and stakeholders.

Part of larger agency - recordkeeping regime reports to:

Multi functional cultural or heritage department

Administrative services department

Library or historical society or museum

Department of State or 'secretariat' equivalent

Office of Chief Executive

4.1.3.1.12.2 Private sector placements

Generally recordkeeping regimes in private enterprises are established to serve the mission of that host body and serve business and regulatory requirements as their first priority. However, most private enterprises now recognise corporate good citizenship as a vital and fragile business asset. As a result, many are relying more on their recordkeeping regimes, particularly their archives, to provide long-term evidence of their societal contributions. Mirroring their organisation's primary evidential concerns, most recordkeeping regimes in the private sector are 'headquartered' in one of the following areas:

- Multi functional administrative services
- Legal department
- Secretariat
- Public relations/Advertising
- Research and development
- Corporate information services, including information systems and information technology

Whether public or private, effective recordkeeping regimes must always be centrally designed and coordinated, but may be decentralised in their implementation and daily operations.

4.1.3.1.13 Type and functional emphasis

The tangible features of the program itself reflect the enterprise that hosts the recordkeeping program. Looking at the categories of human activity below, it is clear that there would be considerable differences in each sector's requirements and use of records. For example, private sector mining is less records intensive than medical services or social welfare.

Categories of human enterprise

Social quality sector enterprises - may be public or private or both.

The recordkeeping program of a fast food company will differ from that of a church; a school will have different records and emphases that a bank and so on.
Some programs serve a single organisation; others collect materials from many different sources. Those who manage the records of a host organisation are referred to as in-house or institutional recordkeeping regimes. Those that receive the inactive records/archives of a number of different bodies are known as collecting archives. Some programs may combine elements of both collecting and in-house work and are characterised as comprehensive recordkeeping regimes.
Increasingly in-house and comprehensive programs are utilising the Records Continuum regime management model to manage records from conception to untimate disposition; whereas collecting programs are more historically orientated and offer repository services exclusively for archival materials. Regardless of whether the type of regime is institutional/in-house, collecting or comprehensive, each one will emphasis different functional aspects of recordkeeping. For example, an institutional regime may be more focussed upon recordkeeping to achieve ongoing business objectives; another might be intent upon addressing legislative and regulatory requirements; a third, usually a collecting regime, might be designed more or less exclusively to recover and preserve evidence of the past. However, it is not unusual for a comprehensive program to be involved in all three, in varying degrees.

Researchers from Monash University's Records Continuum Research Group have developed a checklist of features that characterise a fully competent and effective recordkeeping program.
So far the Group has concluded that the components of accountable recordkeeping include:

4.1.3.1.14 Accountable recordkeeping regimes at macro level.

Independent recordkeeping authority with powers adequate to its purpose

fessional standards and best practice promulgated and accepted by society.
mpliant recordkeeping systems at micro level.
eficial alliances with other accountability players and relationships of trust with
ountability stakeholders

Recordkeeping in this small office creates records supporting its daily business on the left side of the room (just out of the picture) and files its completed business records in the cabinets shown on the right, where they are still accessible, but out of the way. To safeguard their vital records and accommodate their statutory retention requirements, the managers utilise the longer term and security storage services of their public records repository.
Facts, figures and encouragements for good recordkeeping
Authority, reponsibility and powers
It is vital that the top level of decision making understand and approve the recordkeeping regime, its functions and powers; without such recognition, the regime will be unable to fulfil its organisation-wide responsibilities for ensuring evidence and will thus expose the organisation to unacceptable levels of managerial and regulatory risk. The need for AUTHORITY, RESPONSIBILITY and POWER can be articulated in the following way:
One of the most critical tools that a professional recordkeeper can possess is sufficient AUTHORITY to obtain compliance with his/her regime's policies and procedures. This authority to manage/control is granted by the ultimate decision-making entity within an organisation or bureaucratic system and is normally embodied in legislation(public sector) or in executive /administrative orders at the highest level (private sector).
The allocation of regime RESPONSIBILITY is twofold involving (i) Functional or operational responsibility to vest full responsibility and control over all aspects of

records making, using and keeping within the organisation's entire scope of operations in the recordkeeping regime and (ii) Managerial authority which involves giving the recordkeeping professional the authority for developing/implementing/revising/enforcing requirements, standards and guidelines (policies, procedures) for all activities/resources which influence the quality and quantity of records throughout the entire management continuum of records making and keeping.

The POWER to carry out recordkeeping responsibility is embodied in regulations/administrative arrangements, standards, policies and guidelines issued by the recordkeeping regime acting on behalf of the highest authority. These documents specify what can and cannot be done across all activities which comprising and affecting the capture, preservation and accessibility of essential evidence as records. Ideally, all proposed actions impacting upon the quality or quantity of records media, file components, file housings, records creation, records storage areas, records creating/processing/storing technology, reprographics, records and files identification, documentation, maintenance and handling, records access & retrieval systems, automation, disposal, destruction, surveys, vital records protection, etc. should be subject to a coordinated approval and review process.

Unfortunately 1 and 2 in some settings are treated as if they were separate matters, rather than the two halves of the management whole. In such cases, an early and strong effort must be made to reunite and balance them; otherwise your regime cannot not be effective.

In addition to the powers Ketelaar recommends, there are other measures that can facilitate recordkeeping effectiveness - see Powers assisting recordkeeping effectiveness.

Counting and accounting: Communicating the 'value' of recordkeeping

One of the difficulties alluded to by many recordkeepers is communicating the value of recordkeeping to non-specialist stakeholders, particularly to those who pay the bills and expect results and value for money. The challenge is to measure the impact of recordkeeping activity BEFORE and AFTER an interval and then express the outcomes in terms that the target audience values.

Creating and caputuring the information embodied in records represents a considerable investment. One has only to look at the salaries of those responsible for preparing and analysing high level management documents and reports to see that these items have cost a bundle. In addition, there are those vital databases of client informmation, product inventories, design specifications and plans, If these were lost or could only be found with a great amount of time and effort, the resulting cost in terms of delay would be huge.

4.1.3.1.15 **Recordkeeping rewards**

It is no surprise that progressive public and private organisations are realising that ONLY a well-managed recordkeeping program or regime provides the full, accurate and trustworthy evidence needed for optimum rewards such as:

1. Management decision support
2. Compliance with legislative/regulatory requirements
3. Risk management, litigation protection and support
4. Organisational continuity, efficiency and productivity
5. Corporate knowledge base quality control and vital asset protection
6. Fountainhead of societal conscience and memory

4.1.3.1.16 **Recordkeeping and related professions**

While recordkeeping institutions are responsible for capturing and maintaining the documentary evidence important to society, they share the overall role of knowledge preservation and cultural transfer with other heritage management institutions including libraries and museums.

4.1.3.1.16.1 **What's the difference?**

Chart 1: The institutions and their holdings compares the traditional professional responsibilities of registries, archives, libraries and museums

Chart 2: Access to facilities and holdings compares the institutions differing levels of user access.

Chart 3: Professional staff and issues compares the qualifications and professional concerns of staff at registries, archives, libraries and museums.

Most people base their expectations of original sources and archival research on early experiences at school and their use of local public libraries.
Many are surprised that they cannot 'browse' the shelves and borrow originals of records for use at home. They soon understand the need to protect irreplaceable materials by using facsimile copies.

4.1.3.1.17 **Recordkeeping and specialist recordkeepers**

The activities of fully mature recordkeeping regimes document the present and reconstruct the past; they serve business and culture equally through work in offices and in repositories. Overall, the professionally educated recordkeeping specialists working in them are competent to perform the Duties of a recordkeeping specialist.
As we have seen almost every process people undertake in the world generates or involves some form of record and EVERYBODY, not just specialists, is involved. We are all de facto recordkeepers, though few of us may be aware of our role as such. However, in this segment, we concentrate on those who undertake the design and management of recordkeeping systems and service regimes as paid professional work. Recordkeeping specialists oversee the infrastructure - the principles, standards, policies, plans, guidelines and technologies - and provide the advice and support that enables people in different contexts to have the documentation required to meet their personal, business, regulatory and cultural obligations.

CHAPTER 5

ARCHIVE PROGRAM

When you click for open our program Fig 5.1 will open.than it will ask you username and password .You can see it in Fig 5.2

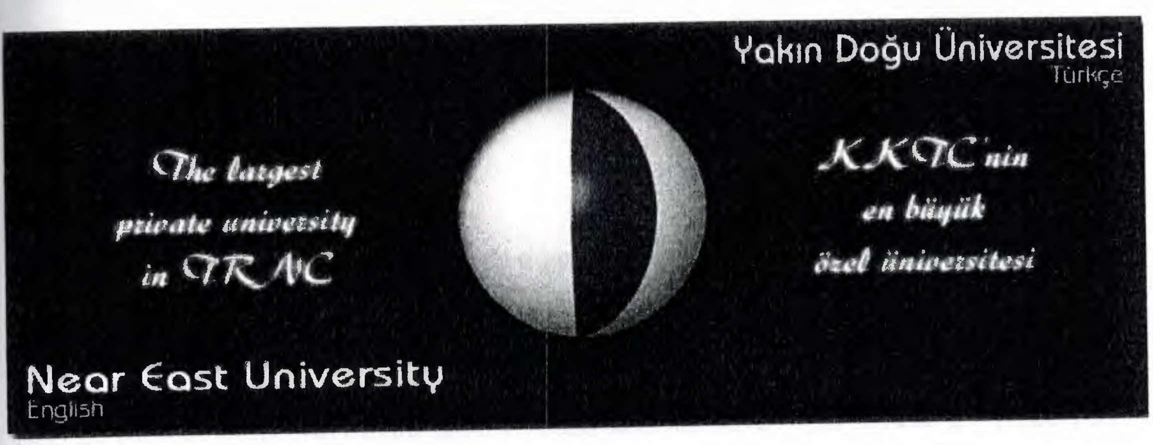

Yakın Doğu Üniversitesi
Türkçe
The largest private university in TRNC
KKTC'nin en büyük özel üniversitesi
Near East University
English

Fig 5.1

If you do not know username and password you can not enter this program.

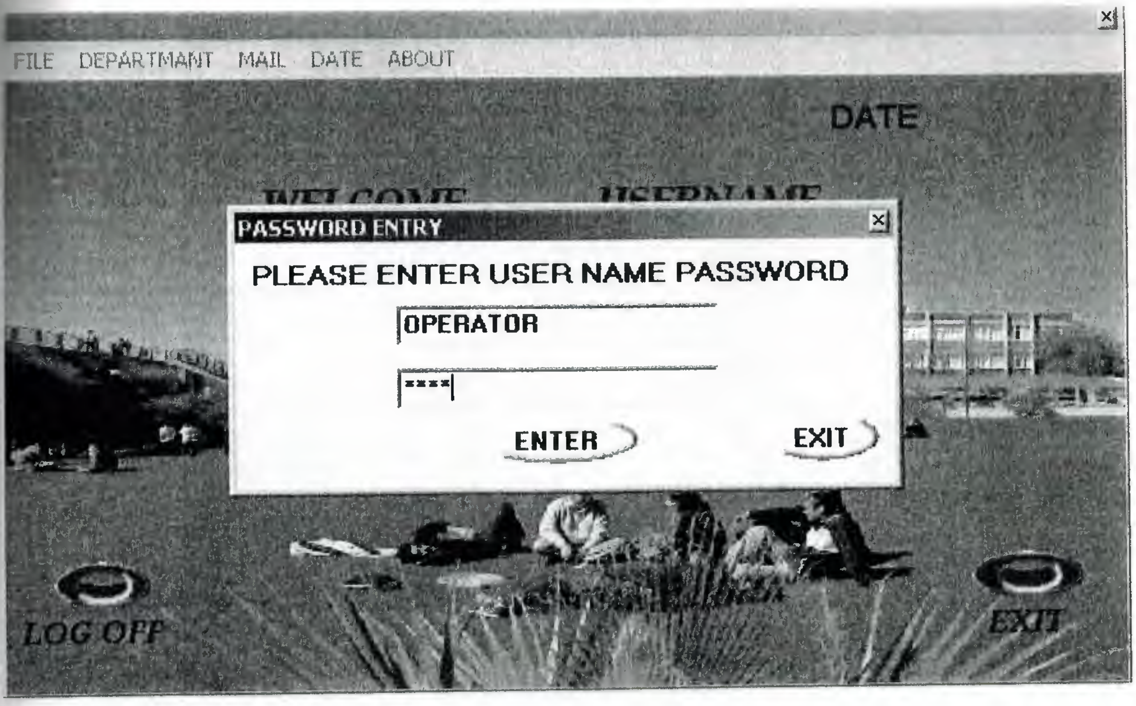


Fig 5.2

In fig5.2 write username and password than click enter for use the archive program.

In our program we have File , Department , Mail , Date and About menus (shown in fig 5.3)

Lets go to start from File menus .In file menu also we have 5 parts these are Received File , Send Files , Reports , List and Close menus.

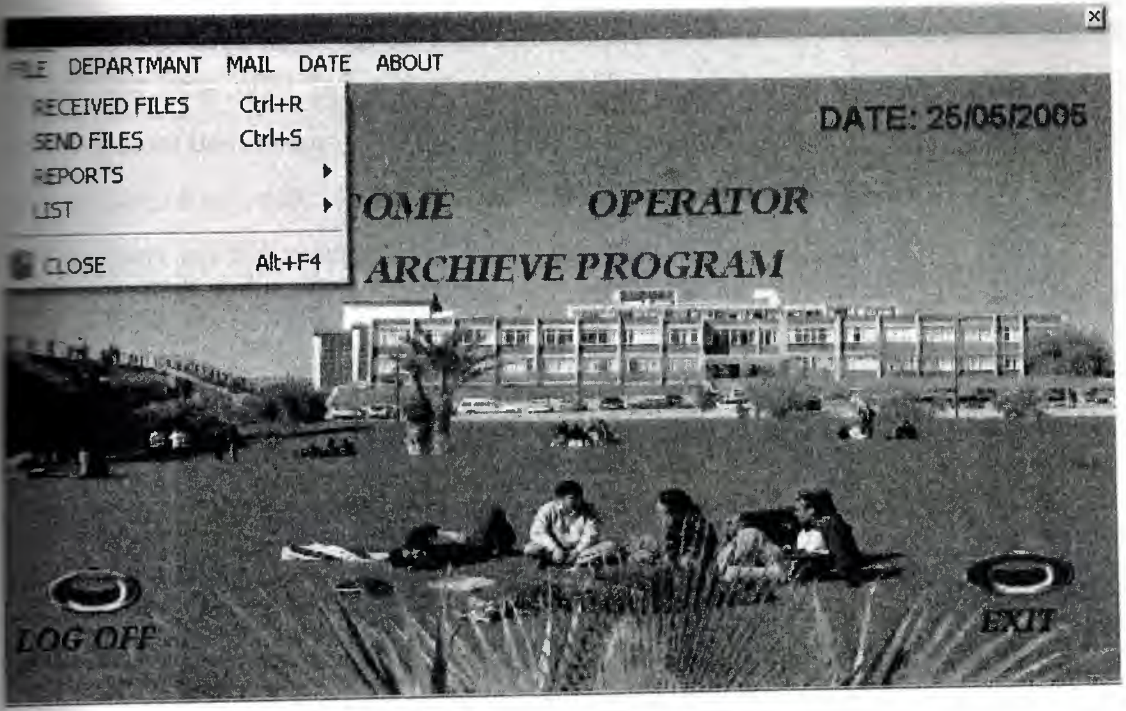
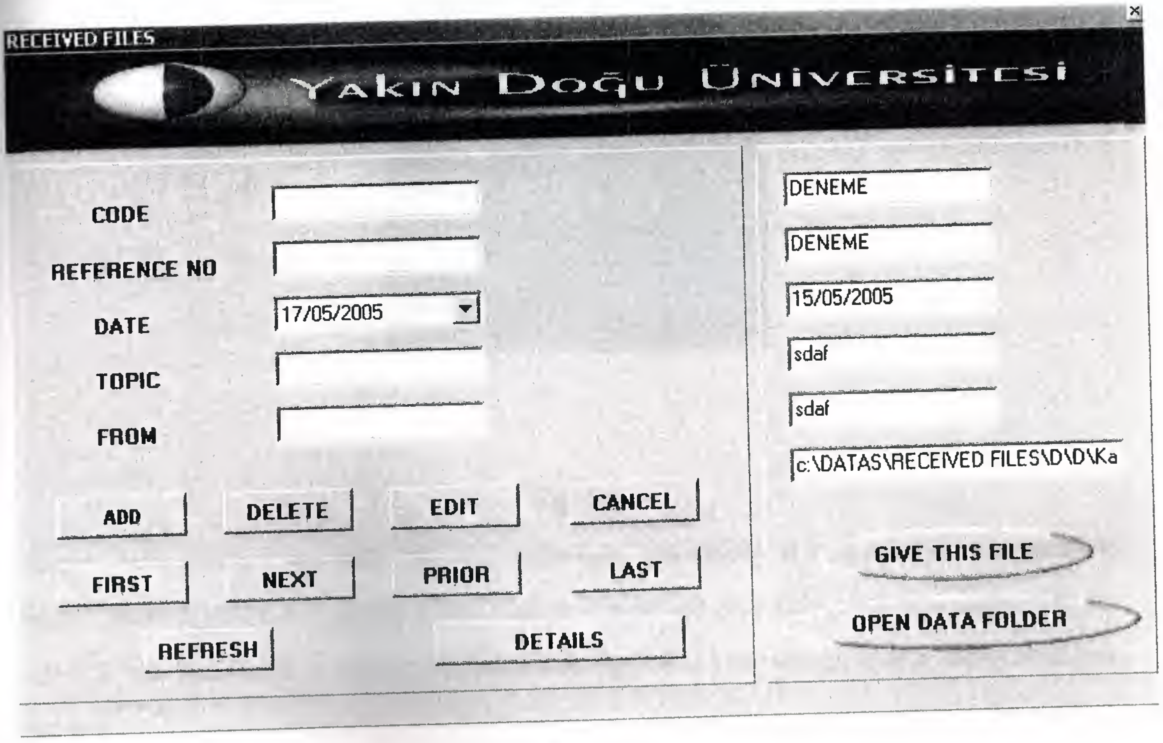


Fig5.3

For Received Files we have to click received files parts or we can use also for short way. We are using this part for working with received documents .Its shown in fig 5.4.

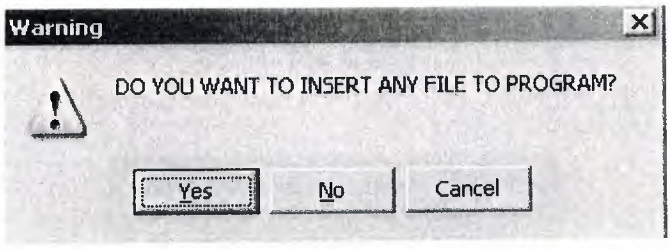


CODE	
REFERENCE NO	
DATE	17/05/2005
TOPIC	
FROM	

DENEME
DENEME
15/05/2005
sdaf
sdaf
c:\DATAS\RECEIVED FILES\D\D\Ka

Fig 5.4

As you see in fig 5.4 in this part we can add , delete and edit .We can see details and we can use first ,next ,prior ,last ,cancel and refresh button .you can understant what they are doing from their name .They are simple to understand. For add we have to write code , reference no , date , topic and from then click to add button than it will ask if we want to insert any file to program .Like Fig 5.5



DO YOU WANT TO INSERT ANY FILE TO PROGRAM?

Fig 5.5

If we want we have to click yes than again it will ask what we want to add word or scan. Shown In fig5.6.If it is word, word documents will open else scan documents will open. If we don't want to add any file we have to click no and save our work in the program.

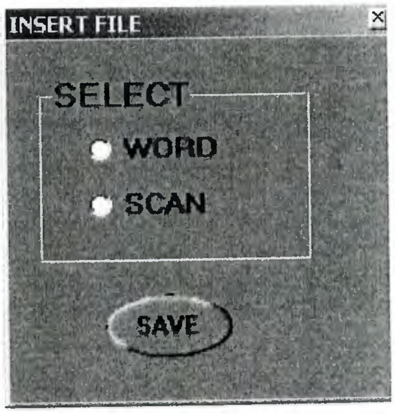
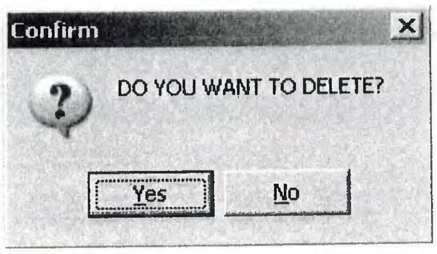


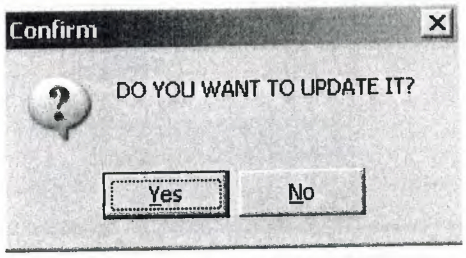
Fig 5.6

Delete button when we will click delete button for delete it will ask "do you want to delete" shown in fig 5.7. If yes it will delete it else not .For Edit button it same also it will ask "do you want to update it" shown in fig 5.8 if yes we can make some changes else not.


Confirm

DO YOU WANT TO DELETE?

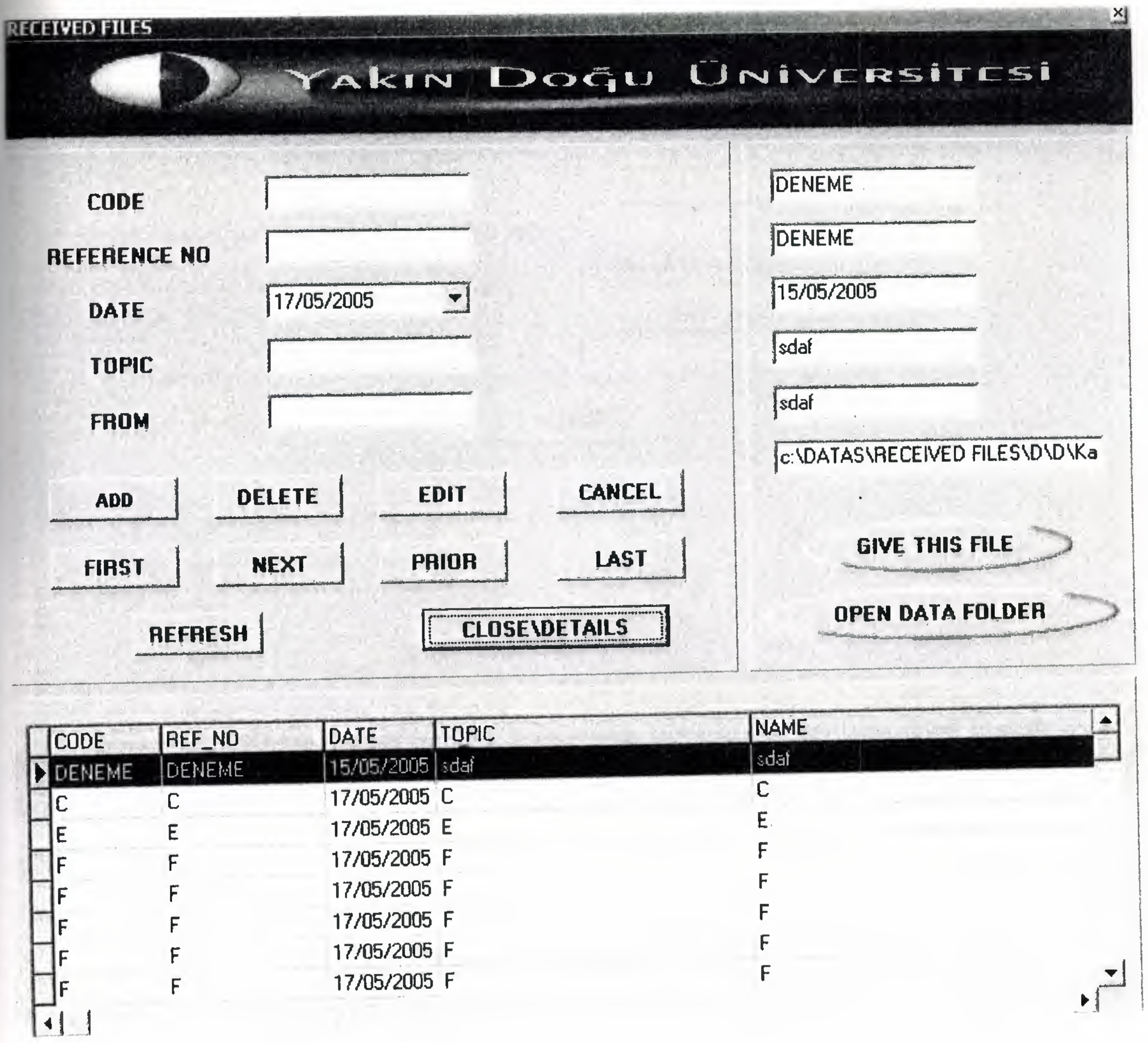
Fig 5.7


Confirm

DO YOU WANT TO UPDATE IT?

Fig 5.8

If we want to see details we have to click detail button. Shown in Fig 5.9 when we click it , it will show details then, button name will change its name to close/detail and if click close detail, button it will close the details and it will change button name to details again .



CODE	
REFERENCE NO	
DATE	17/05/2005
TOPIC	
FROM	

DENEME
DENEME
15/05/2005
sdaf
sdaf
c:\DATAS\RECEIVED FILES\D\D\Ka

CODE	REF_NO	DATE	TOPIC	NAME
DENEME	DENEME	15/05/2005	sdaf	sdaf
C	C	17/05/2005	C	C
E	E	17/05/2005	E	E
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F

Fig 5.9

If we want to send this document to other personel we have to use give this file button. Then it will open new page its shown in Fig 5.10 we will find which one we want to send then it will and send it.

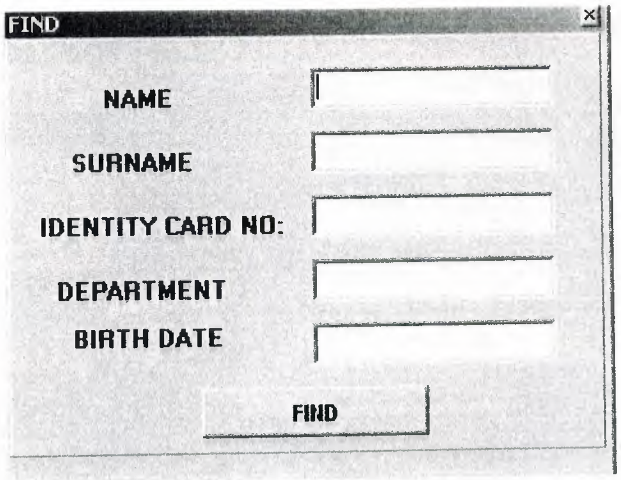


Fig 5.10

About open data folder button, which documents we saved it we can open it with this button .When we click it Fig 5.11 will open. Than we can see the document's .Also it can be scan pages here.

Fig 5.11

For Send Files we have to click Send Files parts or we can use also for short way. We are using this part for working with send document's .Its shown in fig5.12

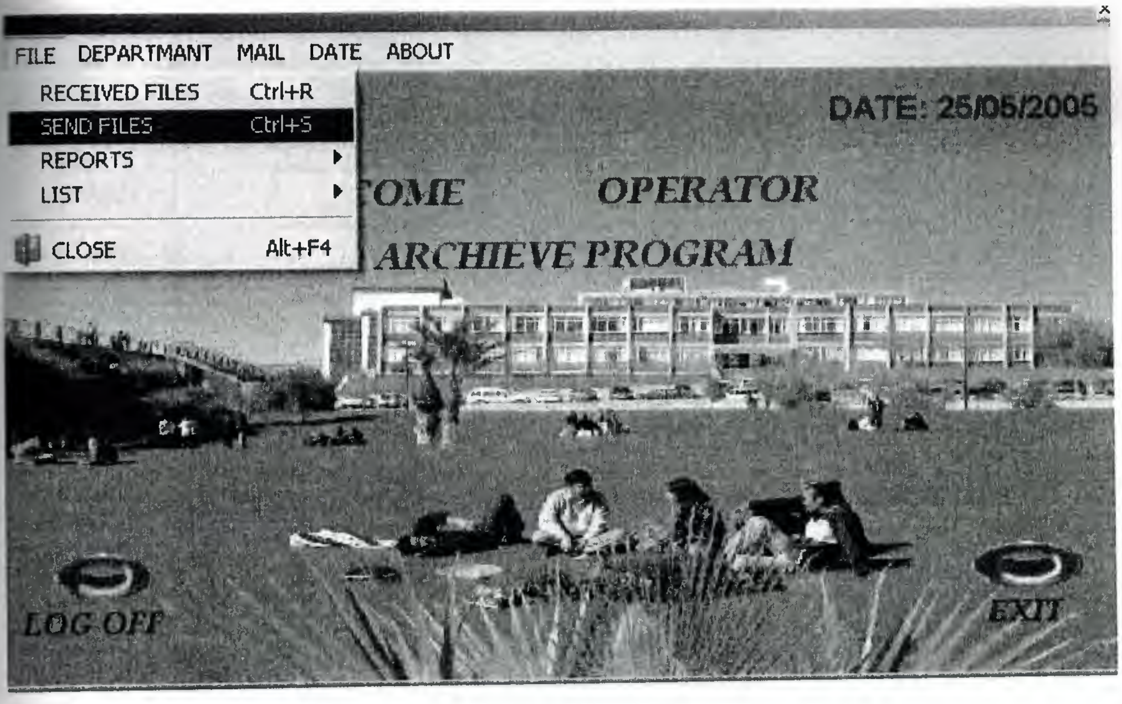
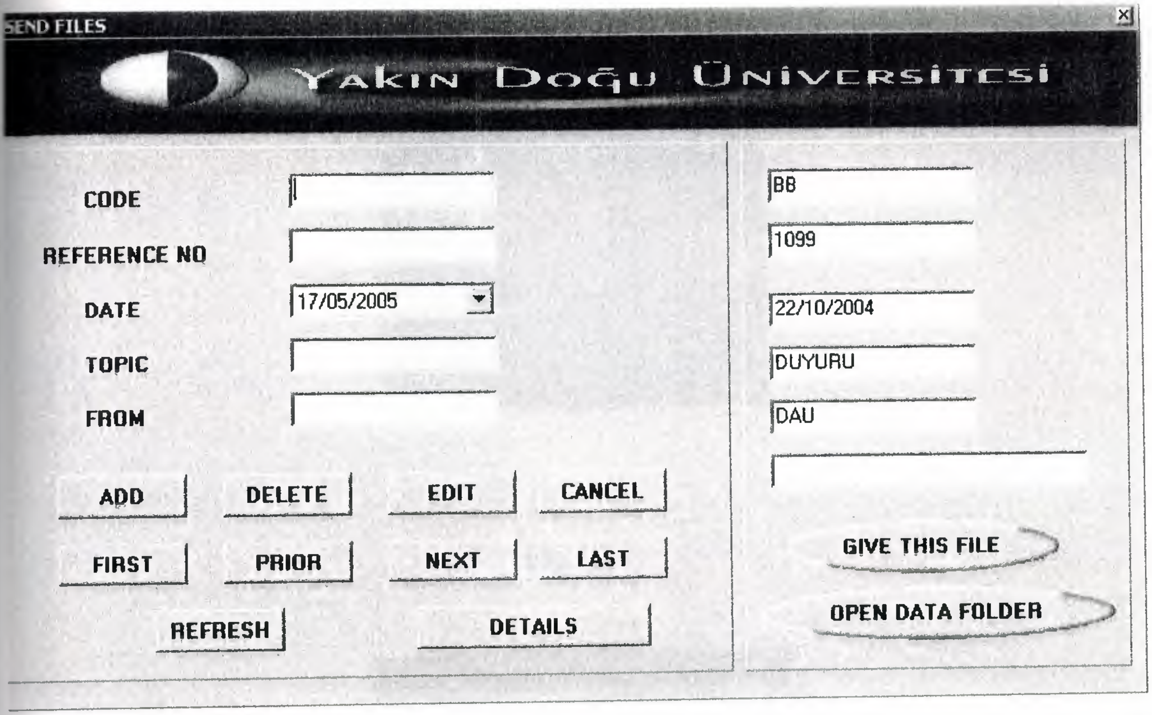


Fig 5.12

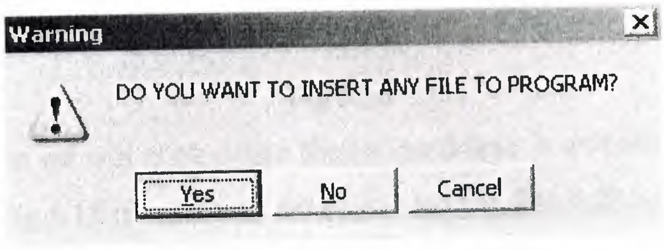
As you see in fig 5.13 in this part we can add , delete and edit .We can see details and we can use first ,next ,prior ,last ,cancel and refresh button .You can understant what they are doing from their name .They are simple to understand. For add we have to write code , reference no , date , topic and from then click to add button than it will ask if we want to insert any file to program. Like Fig 5.14



CODE
REFERENCE NO
DATE 17/05/2005
TOPIC
FROM

BB
1099
22/10/2004
DUYURU
DAU

Fig 5.13



DO YOU WANT TO INSERT ANY FILE TO PROGRAM?

Fig 5.14

As you see in Fig 5.14. If we want we have to click yes than again it will ask what we want to add word or scan. Shown In Fig 5.15. If it is word, word documents will open else scan documents will open. If we don't want to add any file we have to click no and save our work in the program.

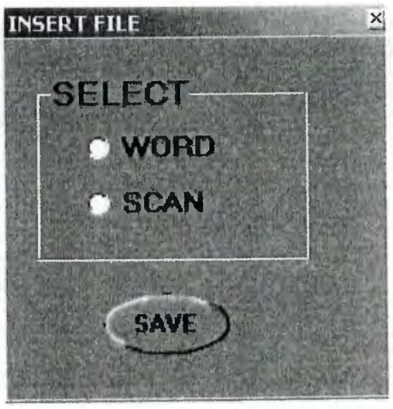
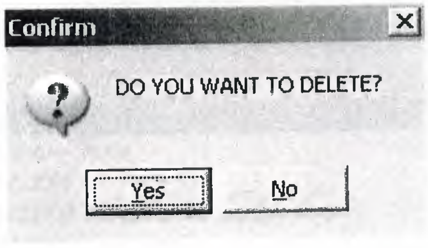


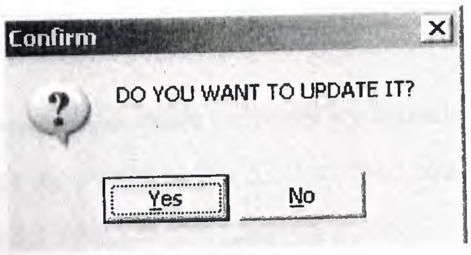
Fig 5.15



DO YOU WANT TO DELETE?

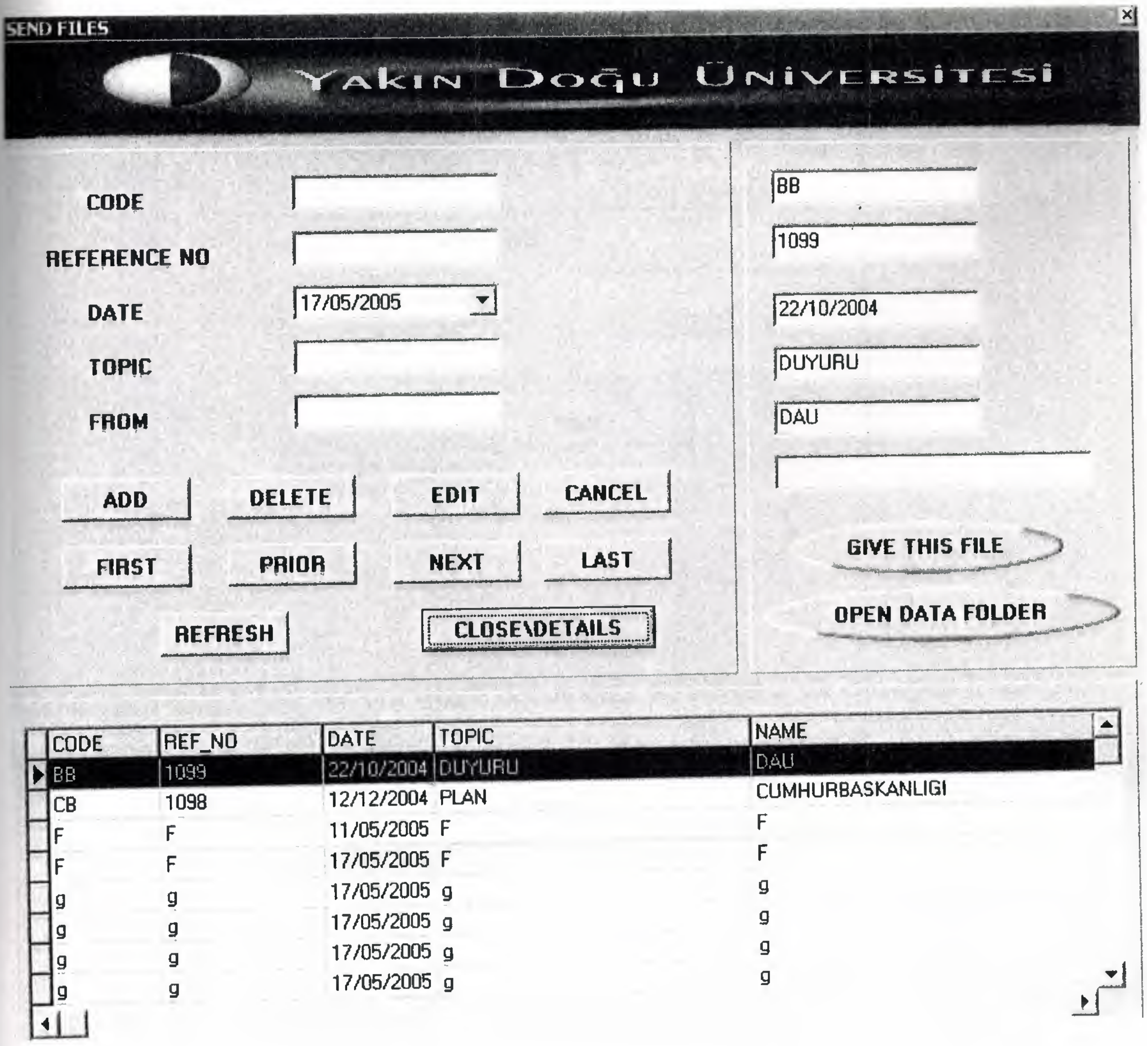
Fig 5.16

Delete button when we will click delete button for delete it will ask “do you want to delete” shown in Fig 5.15 if yes it will delete it , else not .For Edit button it same also it will ask “do you want to update it” shown in fig 5.16 if yes we can make some changes else not.



DO YOU WANT TO UPDATE IT?

Fig5.17



DATE: 17/05/2005

BB
1099
22/10/2004
DUYURU
DAU

CODE	REF_NO	DATE	TOPIC	NAME
BB	1099	22/10/2004	DUYURU	DAU
CB	1098	12/12/2004	PLAN	CUMHURBASKANLIGI
F	F	11/05/2005	F	F
F	F	17/05/2005	F	F
g	g	17/05/2005	g	g
g	g	17/05/2005	g	g
g	g	17/05/2005	g	g
g	g	17/05/2005	g	g

Fig5.18

If we want to see details we have to click detail button. Shown in Fig 5.18 when we click it , it will show details then, button name will change its name to close/detail and if click close detail, button it will close the details and it will change button name to details again .

If we want to send this document to other personel we have to use give this file button. Then it will open new page its shown in Fig 5.19 we will find which one we want to send then it will and send it.

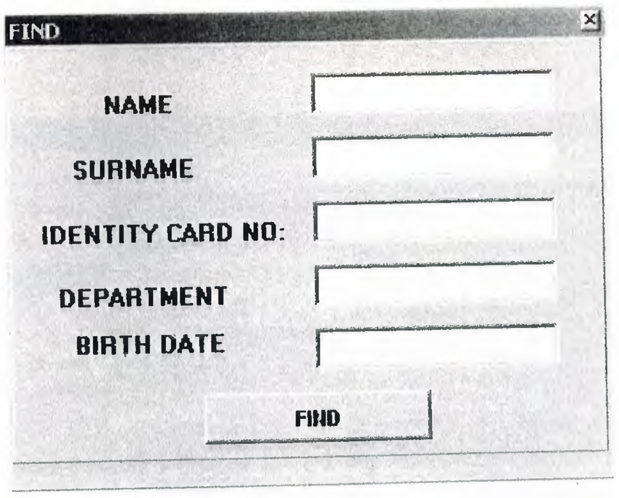


Fig 5.19

About open data folder button, which documents we saved it we can open it with this button.When we click it .It shown in Fig 5.20 than you can open it.

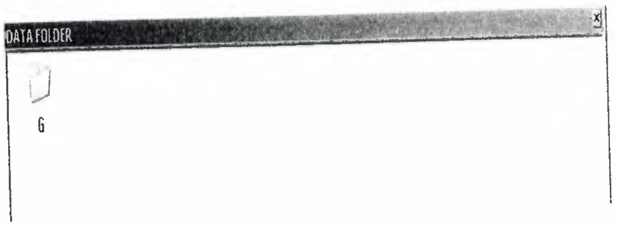
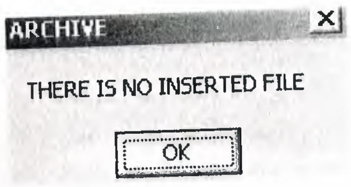


Fig 5.20

Than we can see the document's. Also it can be scan pages here ,but if you haven't any documents saved there it will give us attention shown in Fig 5.21.



ARCHIVE

THERE IS NO INSERTED FILE

OK

Fig 5.21

n the reports we can two more part one for received file and one for send files .It shown n fig 5.21.

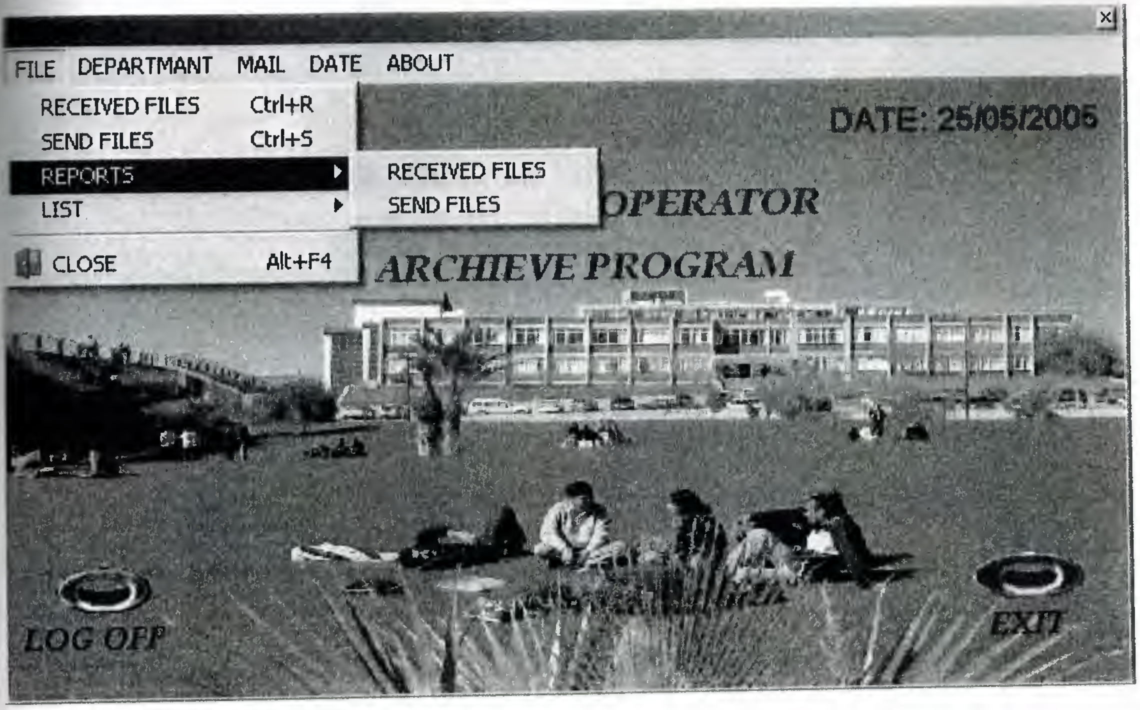


Fig 5.22

When you click received file , its shown in Fig 5.23 we see report for received files .Its shown in fig 5.24.

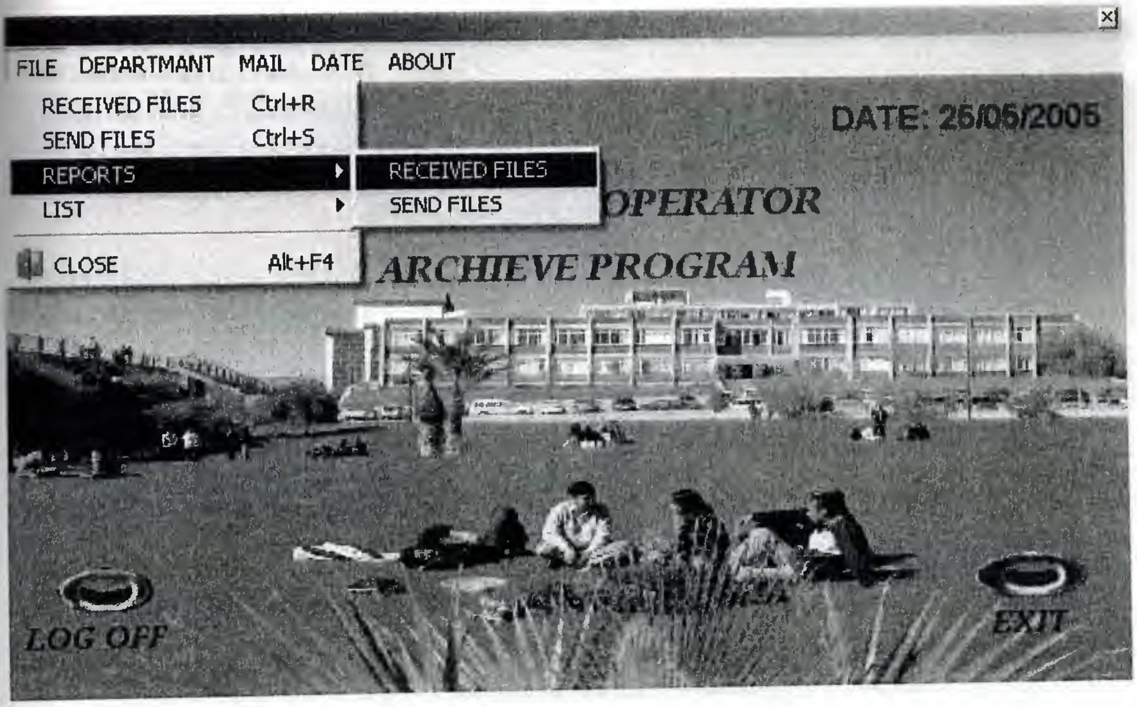
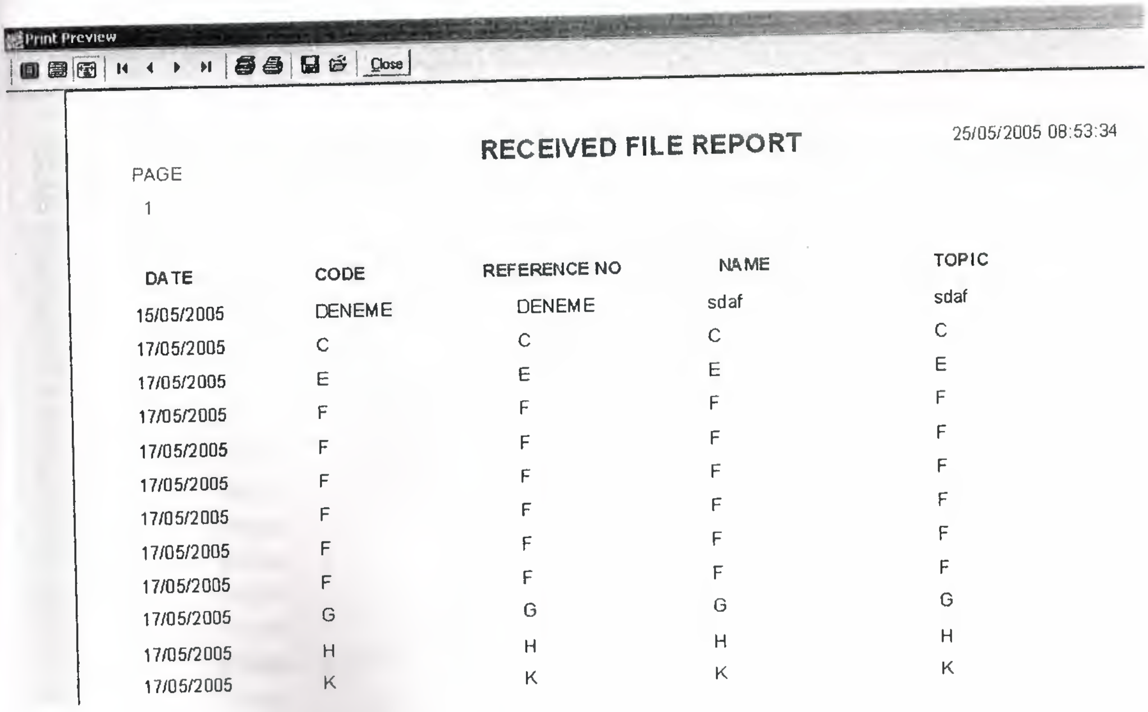


Fig 5.23



RECEIVED FILE REPORT

25/05/2005 08:53:34

PAGE
1

DATE	CODE	REFERENCE NO	NAME	TOPIC
15/05/2005	DENEME	DENEME	sdaf	sdaf
17/05/2005	C	C	C	C
17/05/2005	E	E	E	E
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	G	G	G	G
17/05/2005	H	H	H	H
17/05/2005	K	K	K	K

Fig 5.24

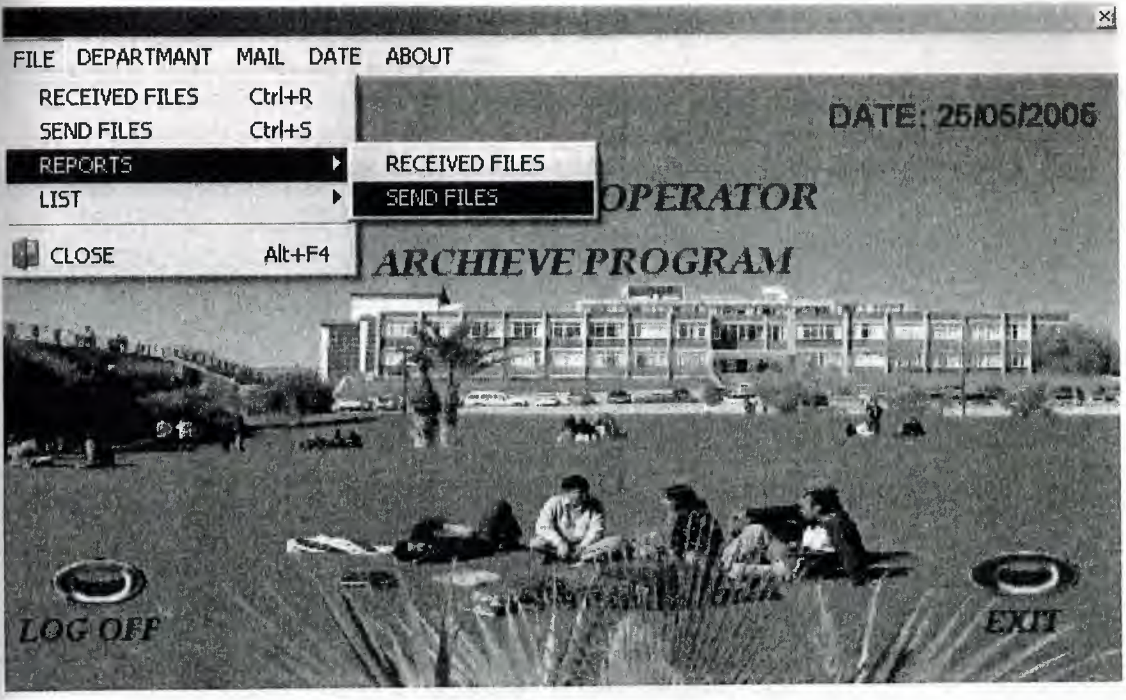
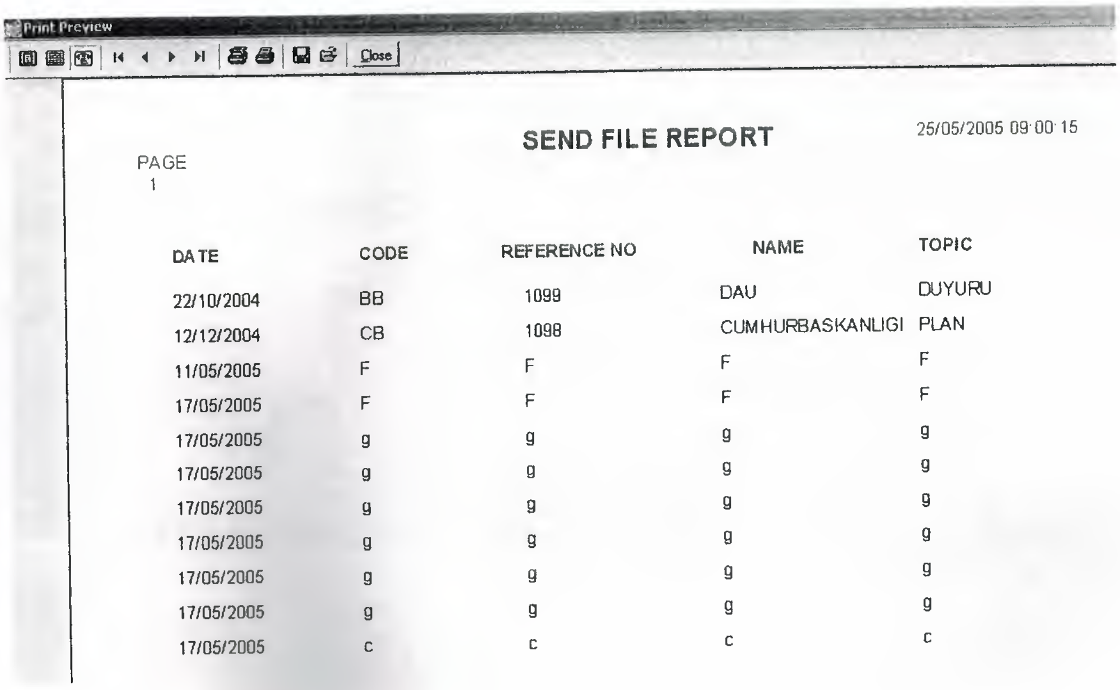


Fig 5.25

If you click send files in reports ,its shown in Fig 5.25 you will see the reports for send files .Shown in Fig 5.26.



SEND FILE REPORT

25/05/2005 09:00:15

PAGE
1

DATE	CODE	REFERENCE NO	NAME	TOPIC
22/10/2004	BB	1099	DAU	DUYURU
12/12/2004	CB	1098	CUMHURBASKANLIGI	PLAN
11/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	g	g	g	g
17/05/2005	c	c	c	c

Fig 5.26

List part its look like reports we can also two parts in this section. These are received file and send files. It's shown in fig 5.27.

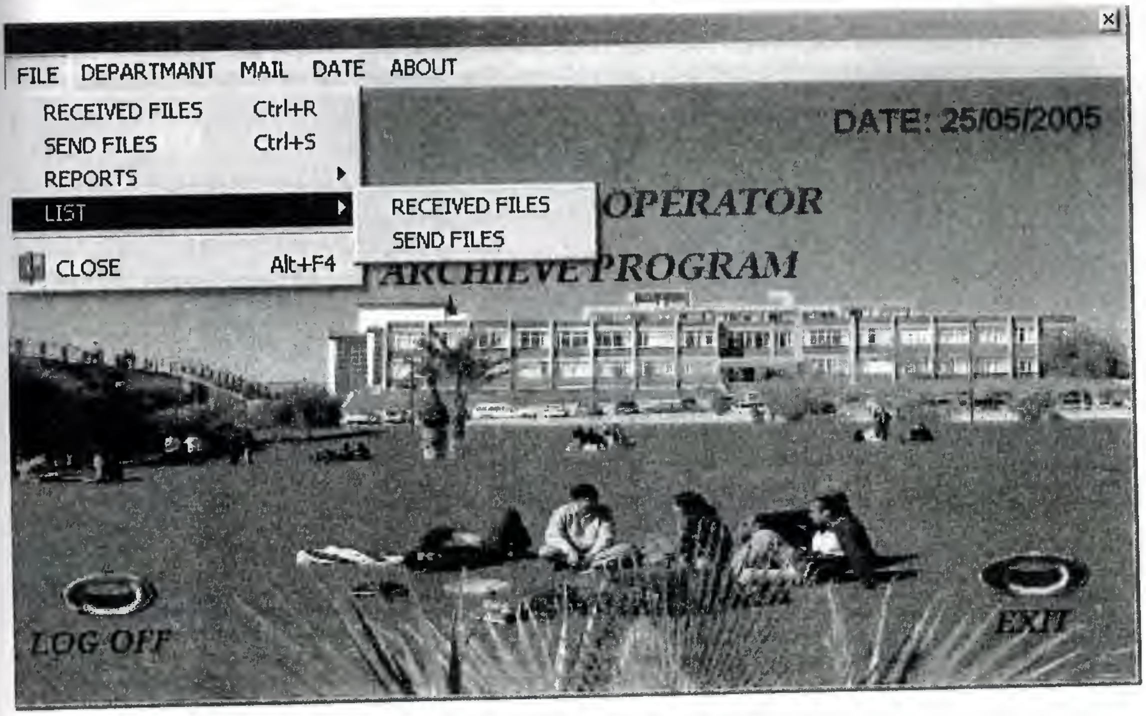


Fig 5.27

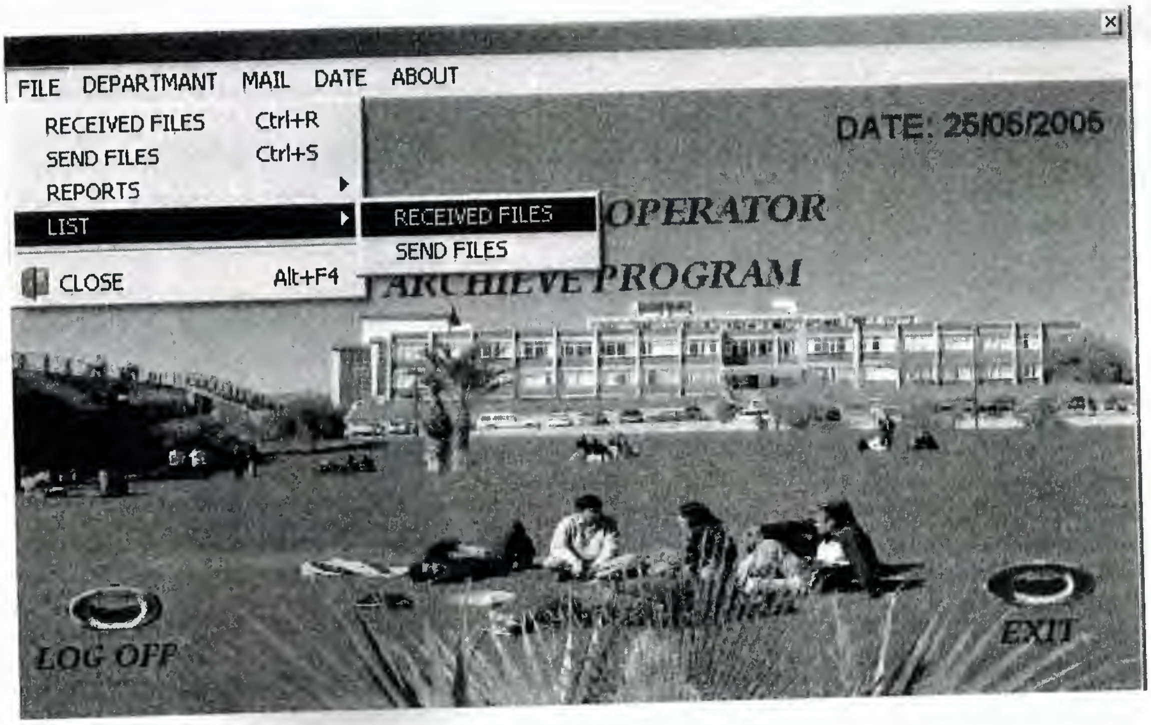


Fig 5.28

If we want see received files we have to click received files in Fig 5.28.When we click it new page will open it its shown in Fig 5.29.

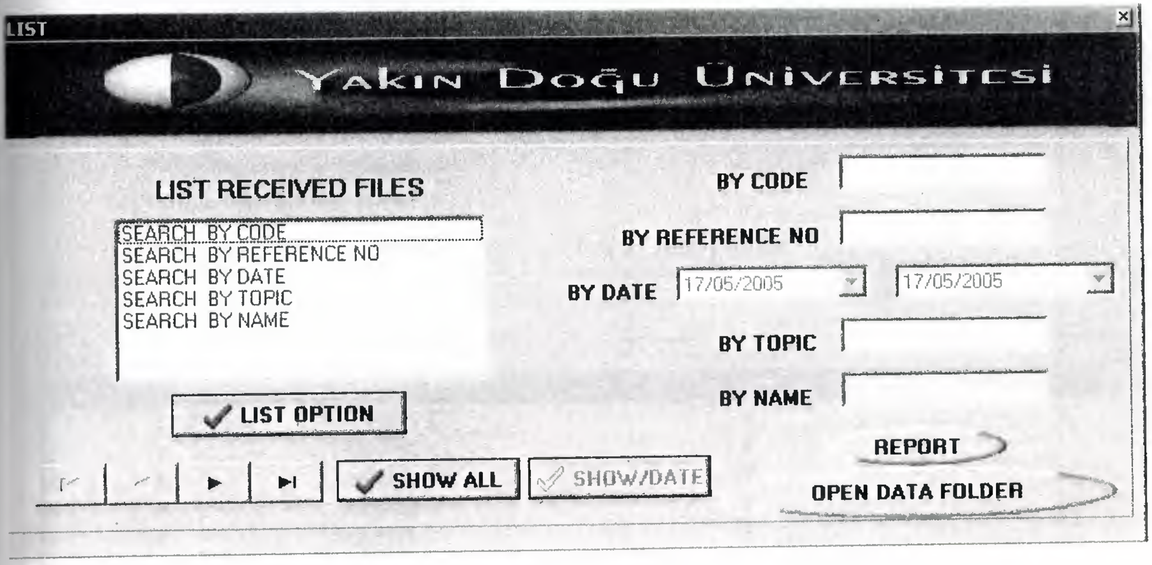


Fig 5.29

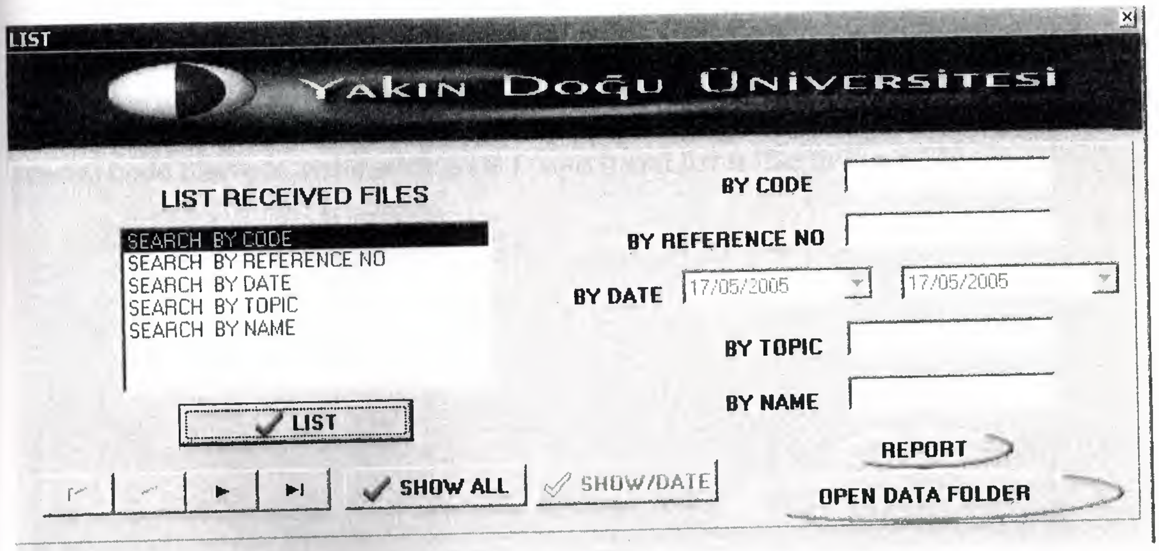
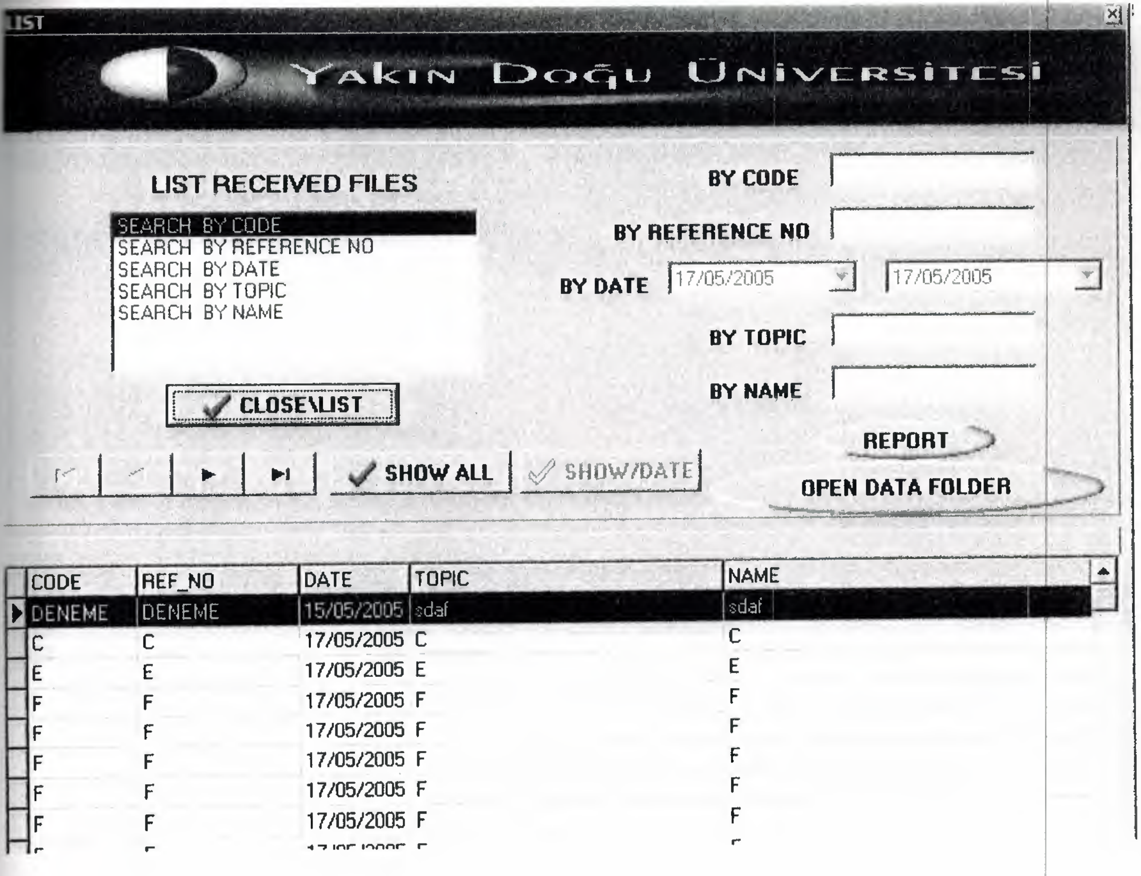


Fig 5.30

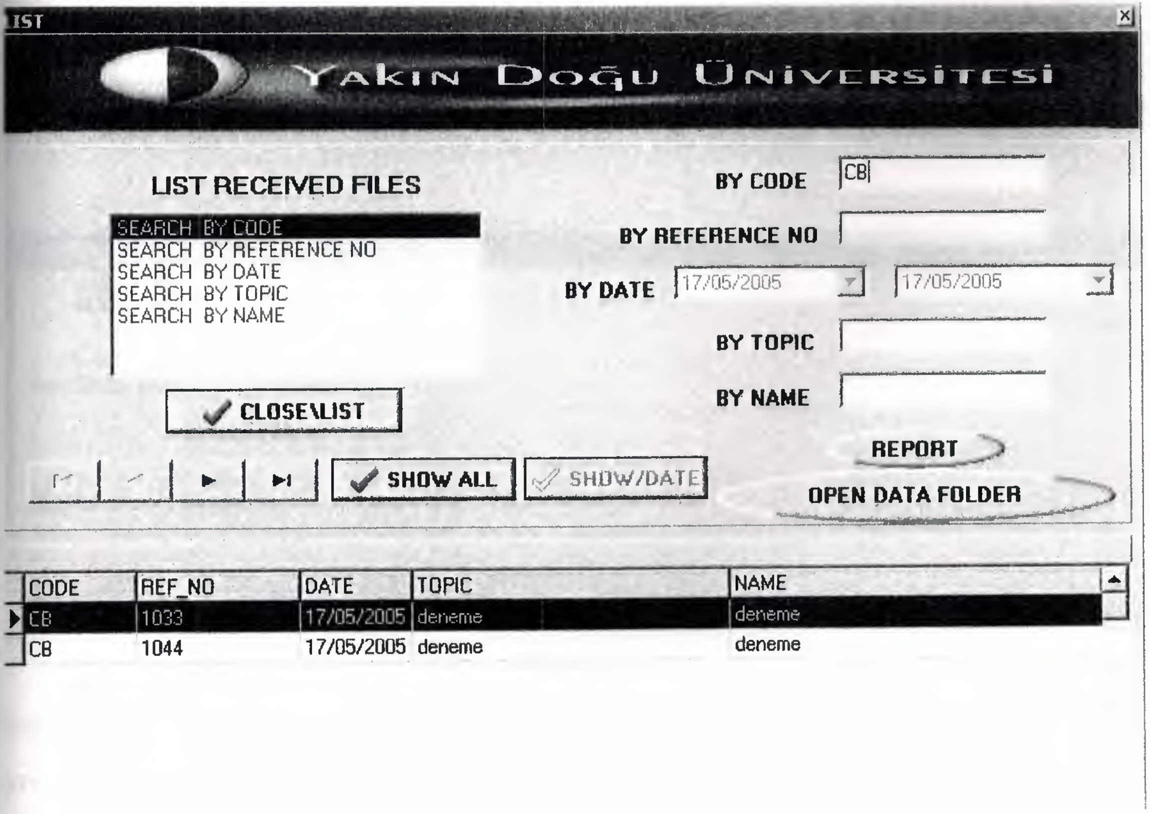
You have to choose which part you want to see its shown in Fig 5.31 and click list when you click it will show you details than list button will change its name to close\list and if you want to close details you can close it with use this button.



CODE	REF_NO	DATE	TOPIC	NAME
DENEME	DENEME	15/05/2005	sdaf	sdaf
C	C	17/05/2005	C	C
E	E	17/05/2005	E	E
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F
F	F	17/05/2005	F	F

Fig 5.31

When you click which part you want for example I choose for search by code for special code I have to write wick code I want it and list it like in Fig 5.32.

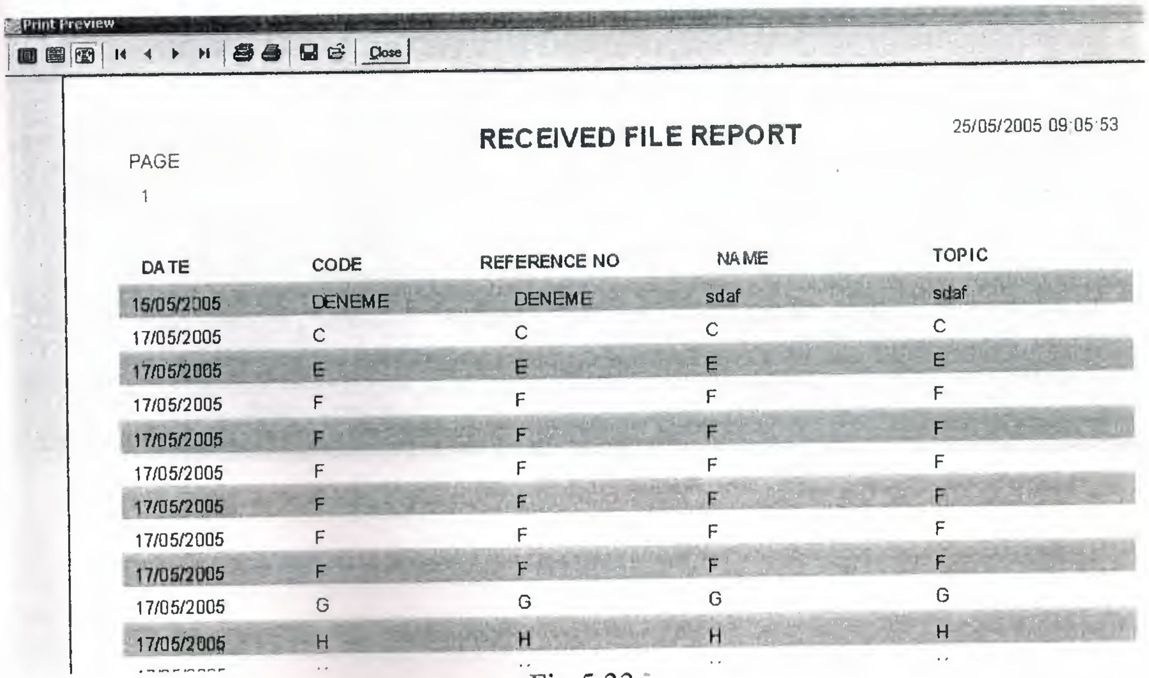


BY CODE: CB
BY REFERENCE NO:
BY DATE: 17/05/2005 17/05/2005
BY TOPIC:
BY NAME:

CODE	REF_NO	DATE	TOPIC	NAME
CB	1033	17/05/2005	deneme	deneme
CB	1044	17/05/2005	deneme	deneme

Fig 5.32

When you click reports button in Fig 5.32 you can see the reports in Fig 5.33.



RECEIVED FILE REPORT

25/05/2005 09:05:53

PAGE
1

DATE	CODE	REFERENCE NO	NAME	TOPIC
15/05/2005	DENEME	DENEME	sdaf	sdaf
17/05/2005	C	C	C	C
17/05/2005	E	E	E	E
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	F	F	F	F
17/05/2005	G	G	G	G
17/05/2005	H	H	H	H

Fig 5.33

If you have some folder to see it in Fig 5.31 when you click the open data folder button you will see your folder its shown in Fig 5.34 and you can open it with double click.

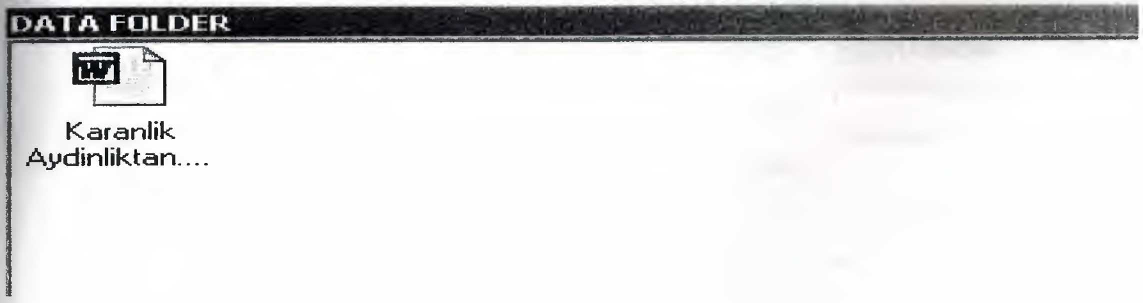


Fig 5.34

For see the send files in list part you have to click send files its shown in Fig 5.35.when you click it , it will open new page for send files its shown in Fig 5.36.

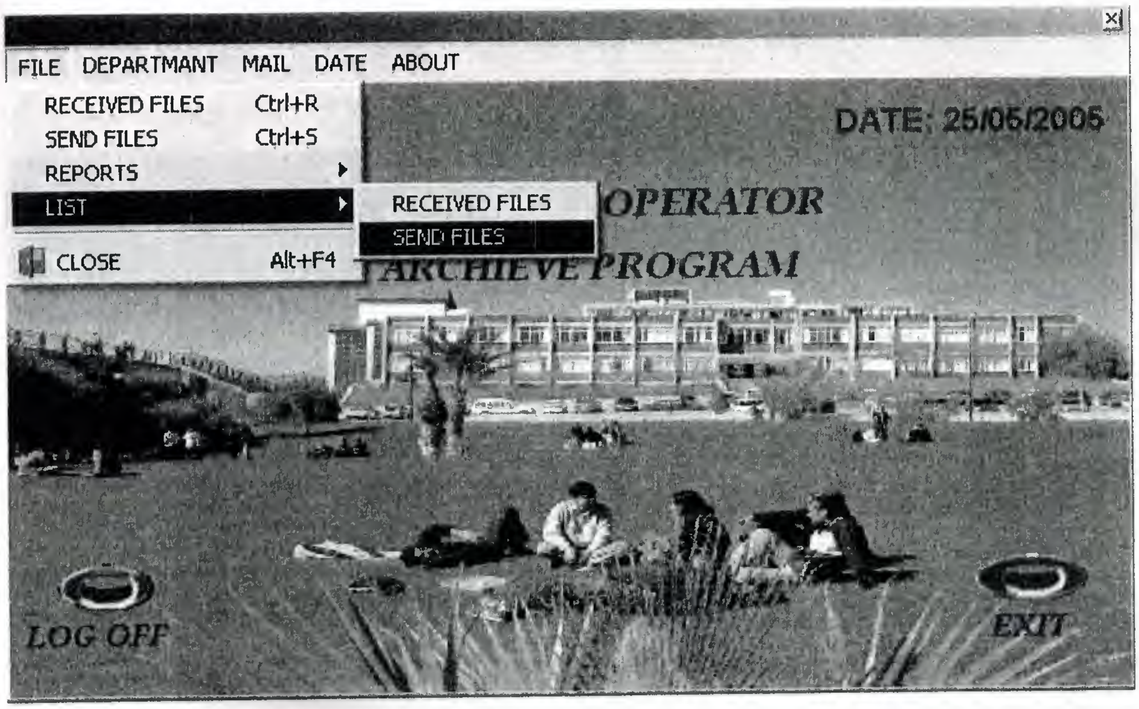


Fig 5.35

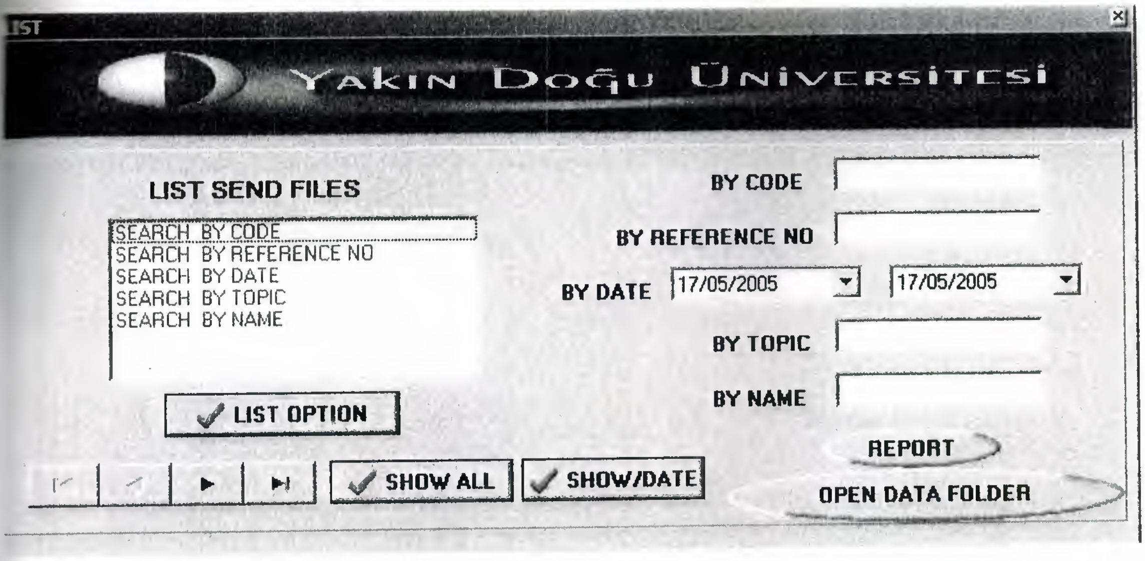


Fig 5.36

Its look like received file you have to choose which part you want to see its shown in Fig 5.37 and click list when you click it will show you details than list button will change its name to close\list and if you want to close detail you can close it with use this button .Its shown in Fig 5.38

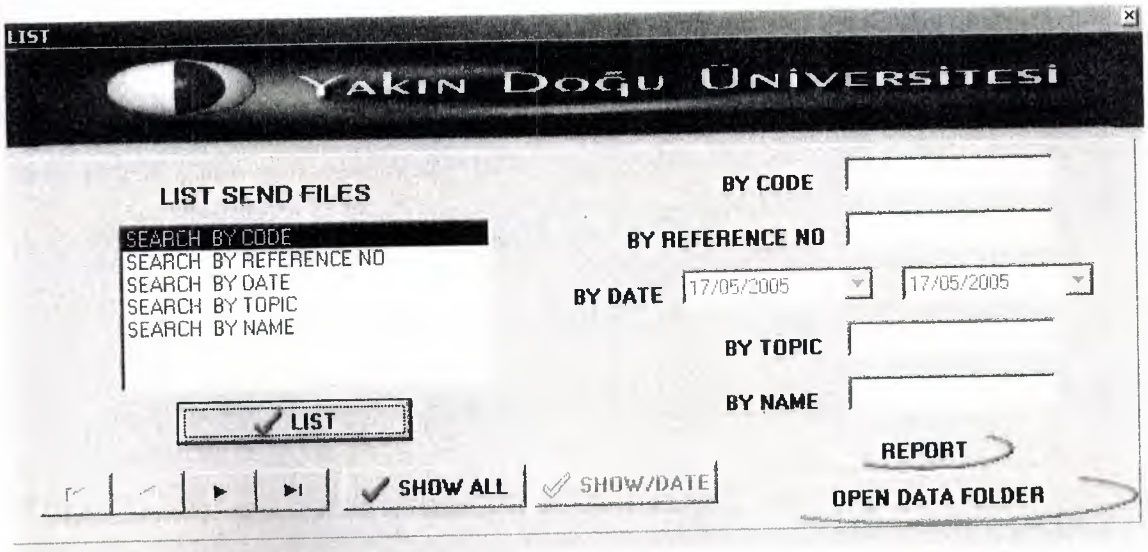
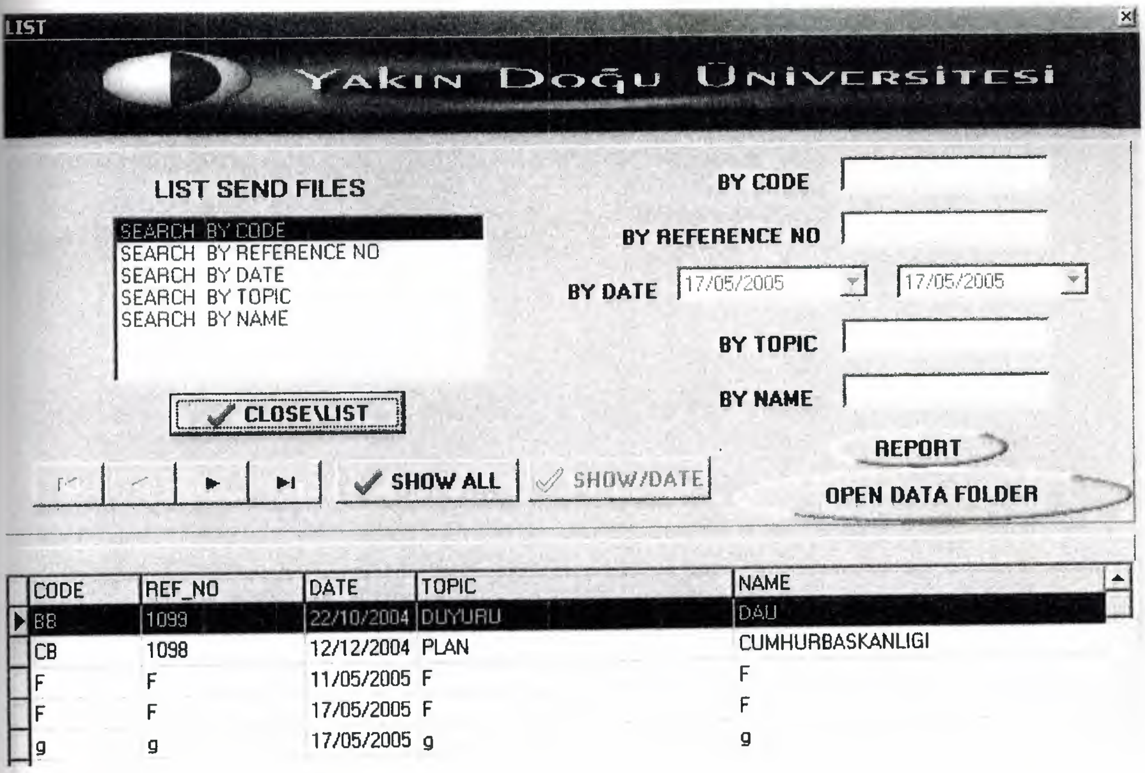


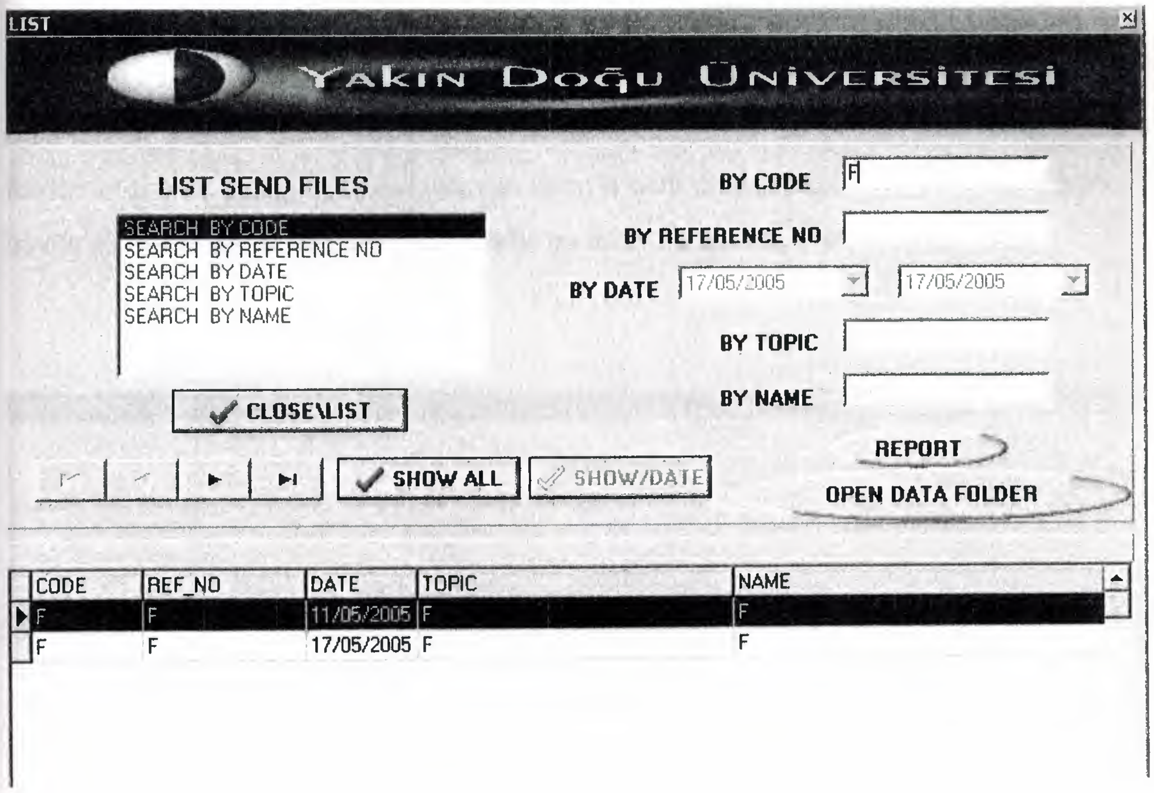
Fig 5.37



CODE	REF_NO	DATE	TOPIC	NAME
BB	1099	22/10/2004	DUYURU	DAU
CB	1098	12/12/2004	PLAN	CUMHURBASKANLIGI
F	F	11/05/2005	F	F
F	F	17/05/2005	F	F
g	g	17/05/2005	g	g

Fig 5.38

When you click which part you want for example I choosed for search by code for special code I have to write wich code I want it and list it like in fig 5.39



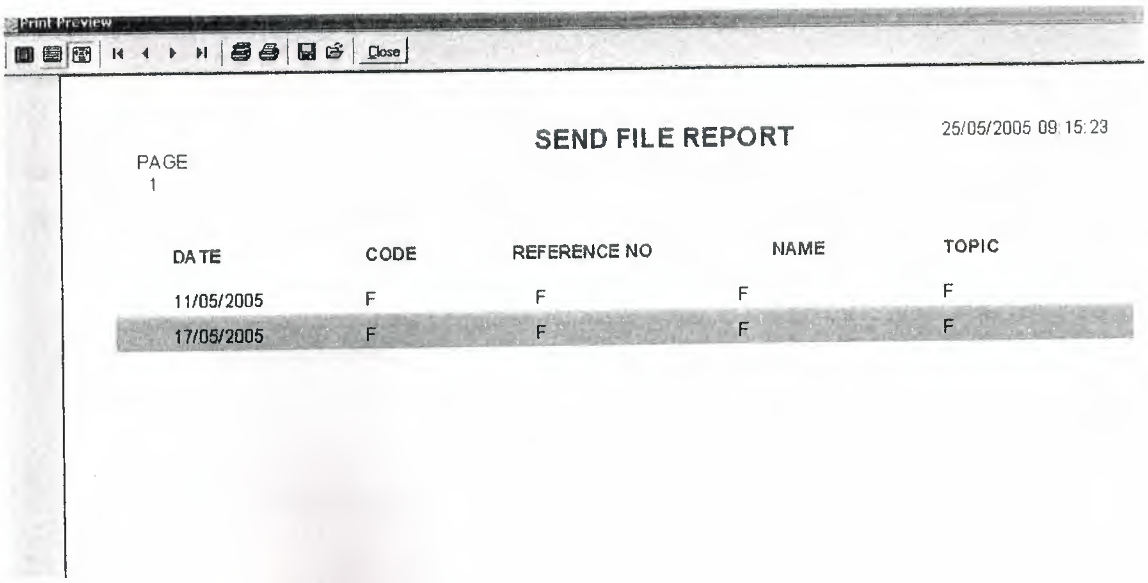
BY CODE: F

BY DATE: 17/05/2005 17/05/2005

CODE	REF_NO	DATE	TOPIC	NAME
F	F	11/05/2005	F	F
F	F	17/05/2005	F	F

Fig 5.39.

When you click reports button in Fig 5.37 you can see the reports in Fig 5.40.



SEND FILE REPORT

25/05/2005 09:15:23

PAGE 1

DATE	CODE	REFERENCE NO	NAME	TOPIC
11/05/2005	F	F	F	F
17/05/2005	F	F	F	F

Fig 5.40

If you have some folder which line you select it when you click the open data folder button it shown in Fig5.37 when you click the open data folder button you will see your folder its shown in Fig 5.41 and you can open it with double click . Else I mean if you haven't any folder it will tell you there is no inserted file its shown in Fig 5.42.

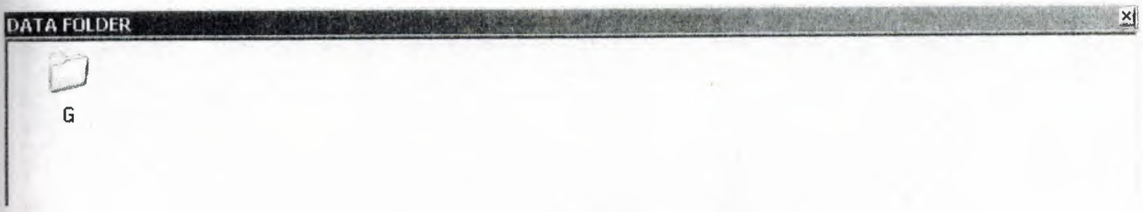
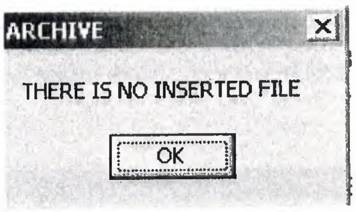


Fig 5.41


ARCHIVE

THERE IS NO INSERTED FILE

OK

Fig 5.42

Last part for file menus is close its shown in fig 5.43 when you will chose this part you will close the archive program.

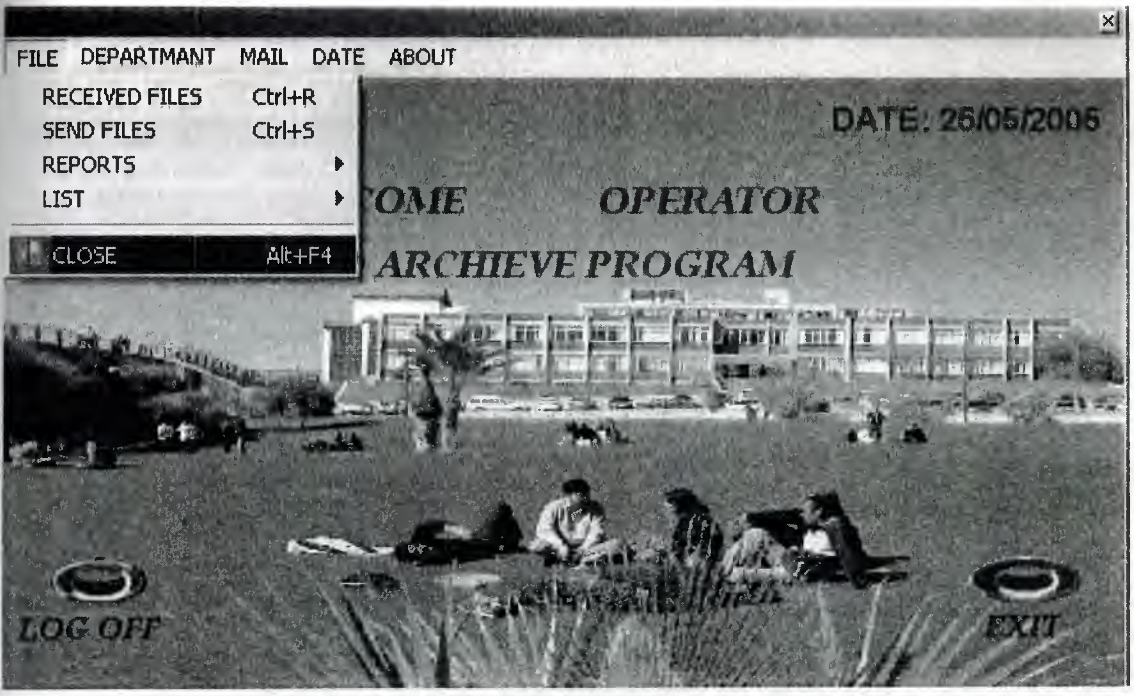


Fig 5.43

Second main menus for our program are department. Department menus have a four part. These are user settings, personal entry, personal list and given files list. As you can see the Fig 5.44.

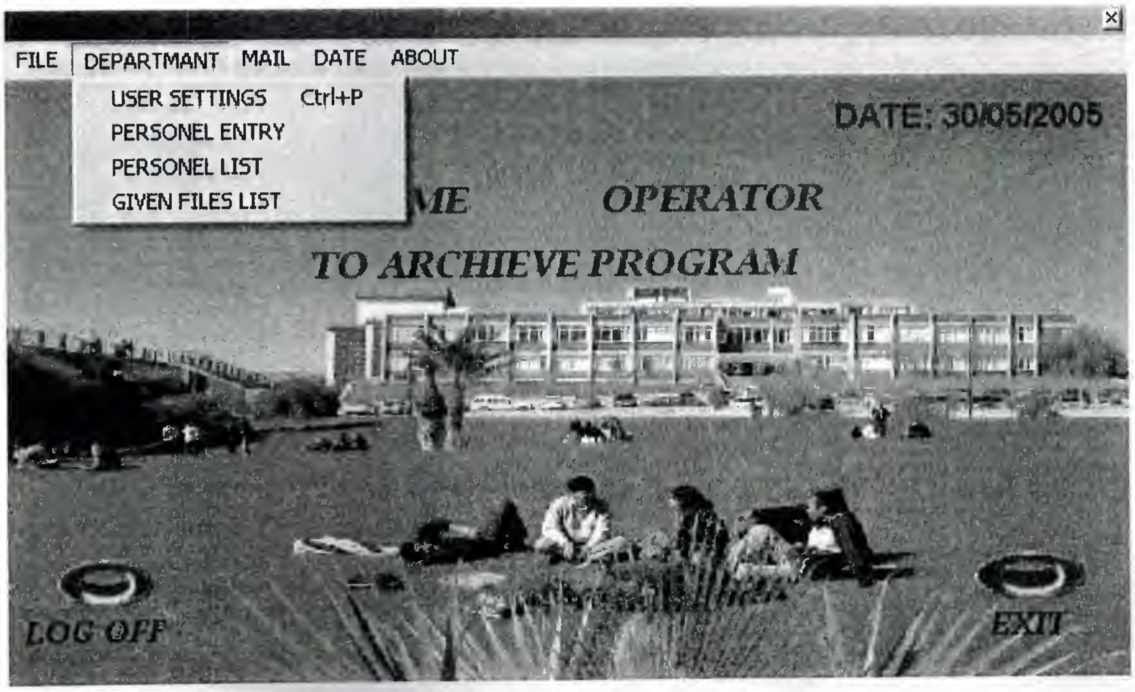


Fig 5.44

for enter the user settings we have to click user settings or we can also use short way to enter this part. Shown in Fig 5.45

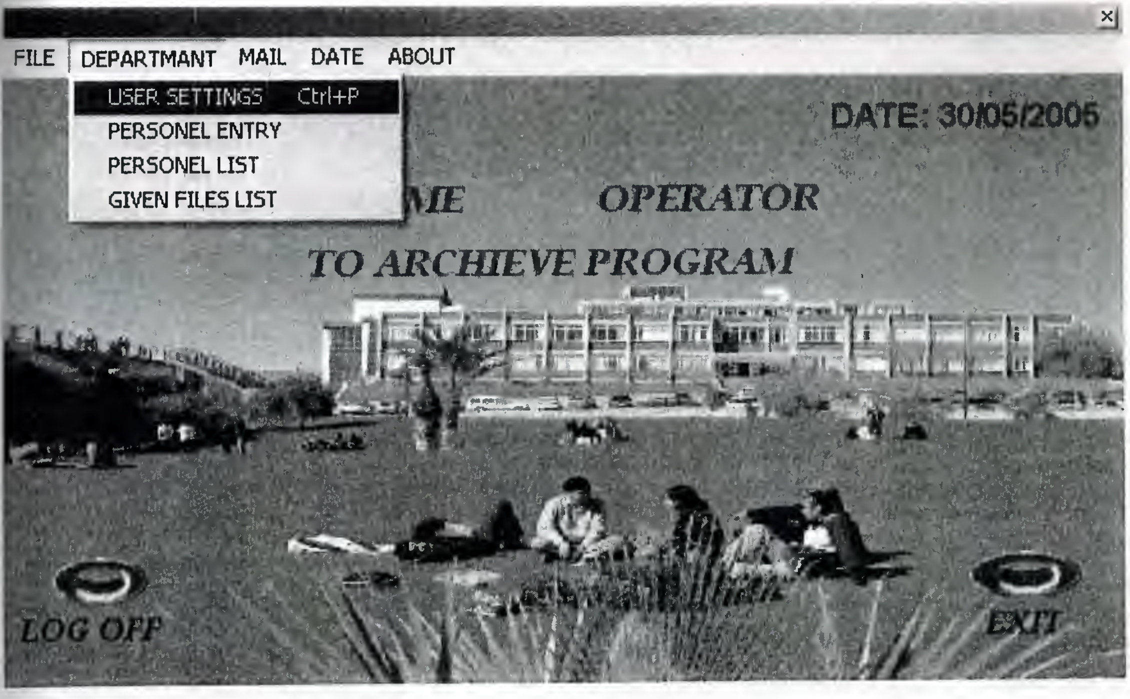


Fig 5.45

When we clicked user settings part it will open new page and it ill ask username and password for enter it shown in fig 5.46

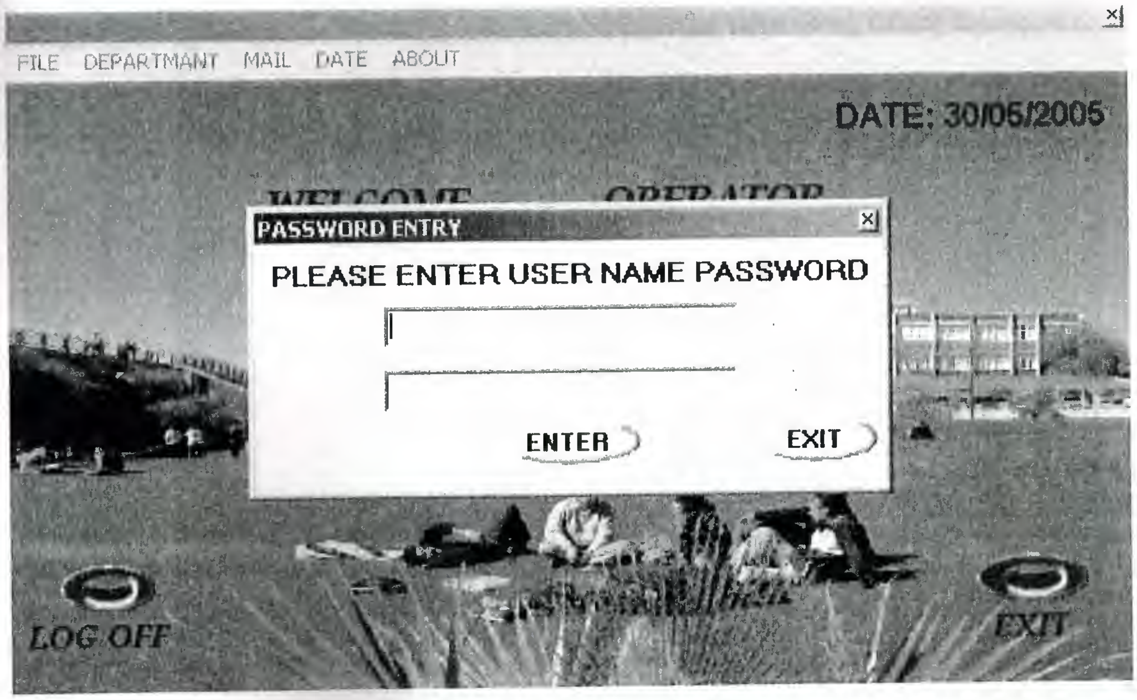
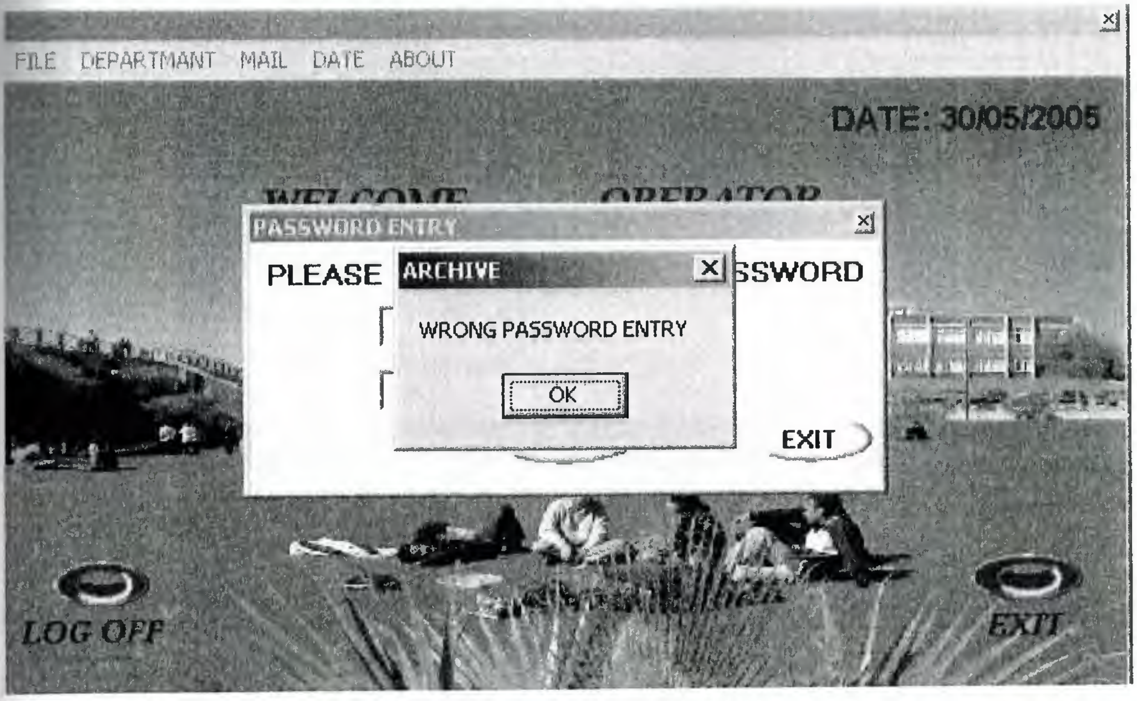


Fig 5.46

If you don't want username and password you can not enter this part shown in Fig 5.47 only the operator can enter this part. Operator is the headmaster of the program .Only operator can do what ever he/she want.



WRONG PASSWORD ENTRY

Fig 5.47

When you write correct username and password it will open user settings pages. In this part you give permissions for the users also you can add new user , delete the users and update the users. As you can see in the Fig 5.48.

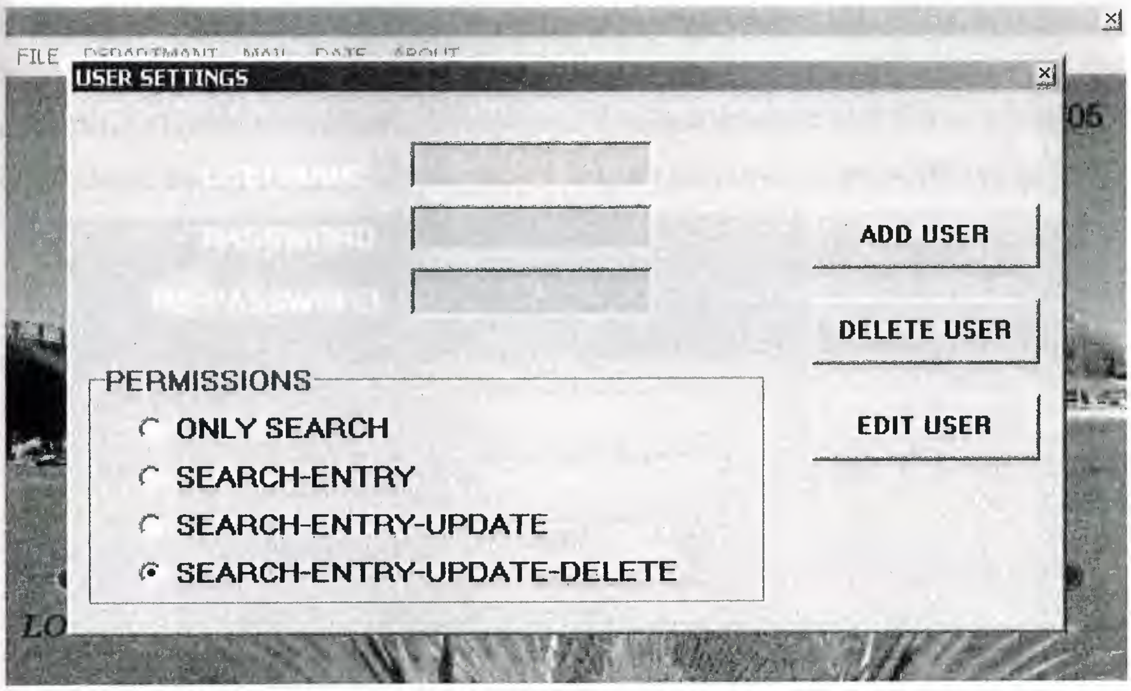


Fig 5.48

Personal entry from the department shown in Fig 5.49.when you click it will open personal entry's page shown in Fig 5.50

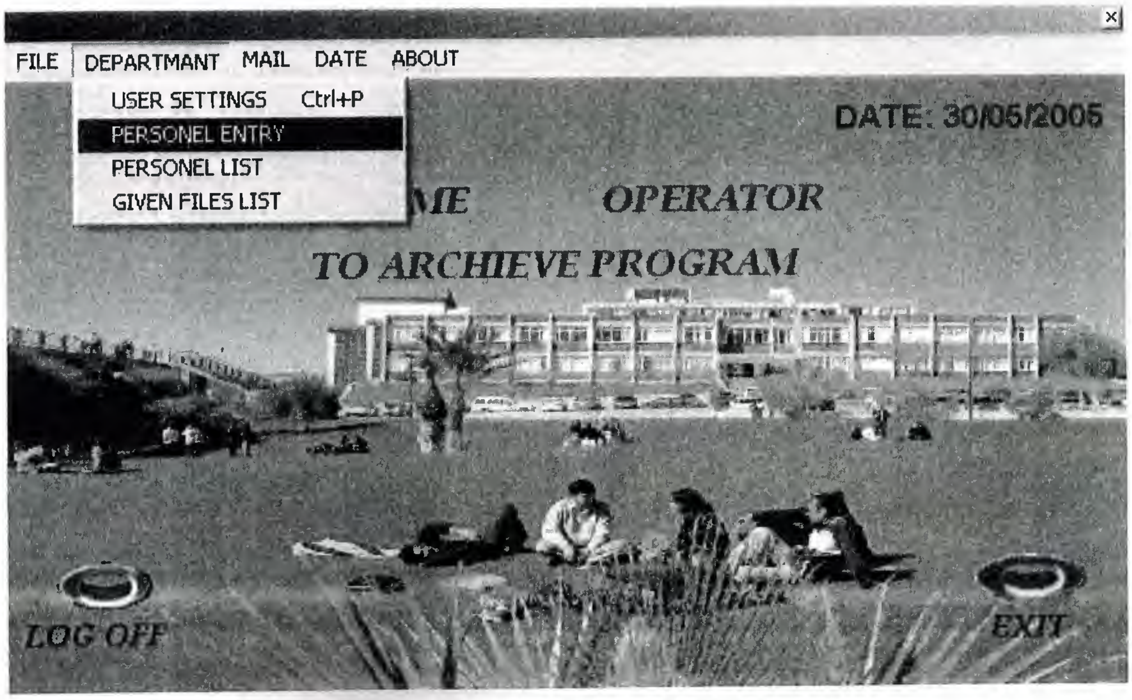
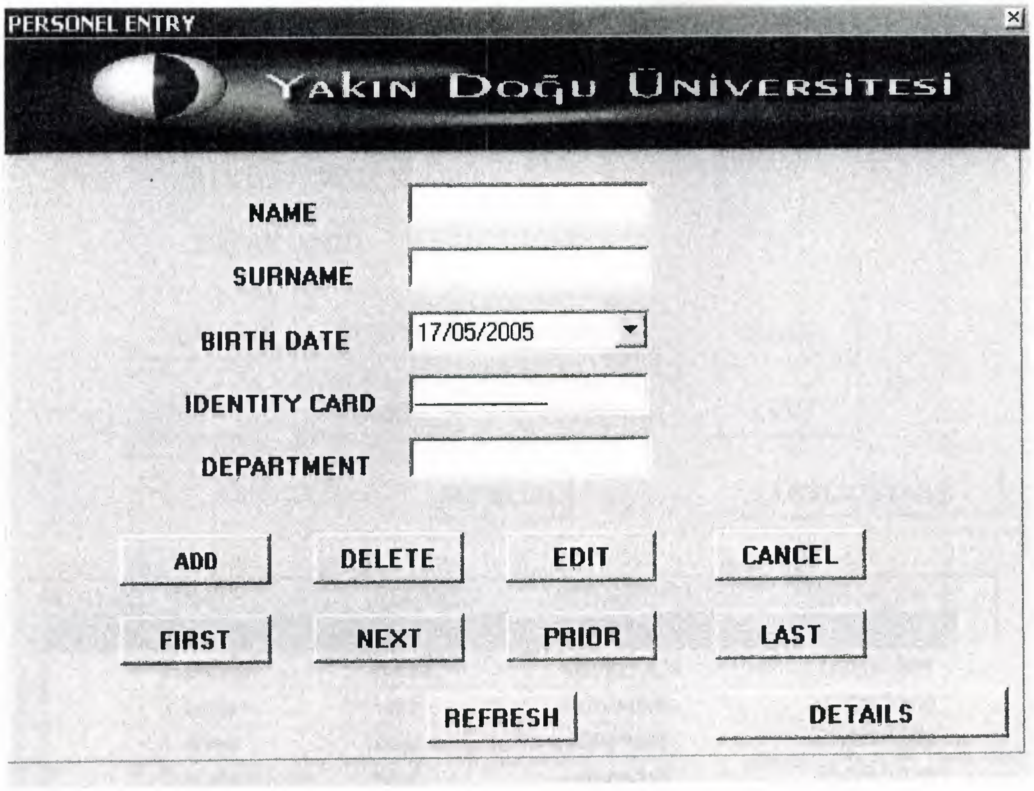


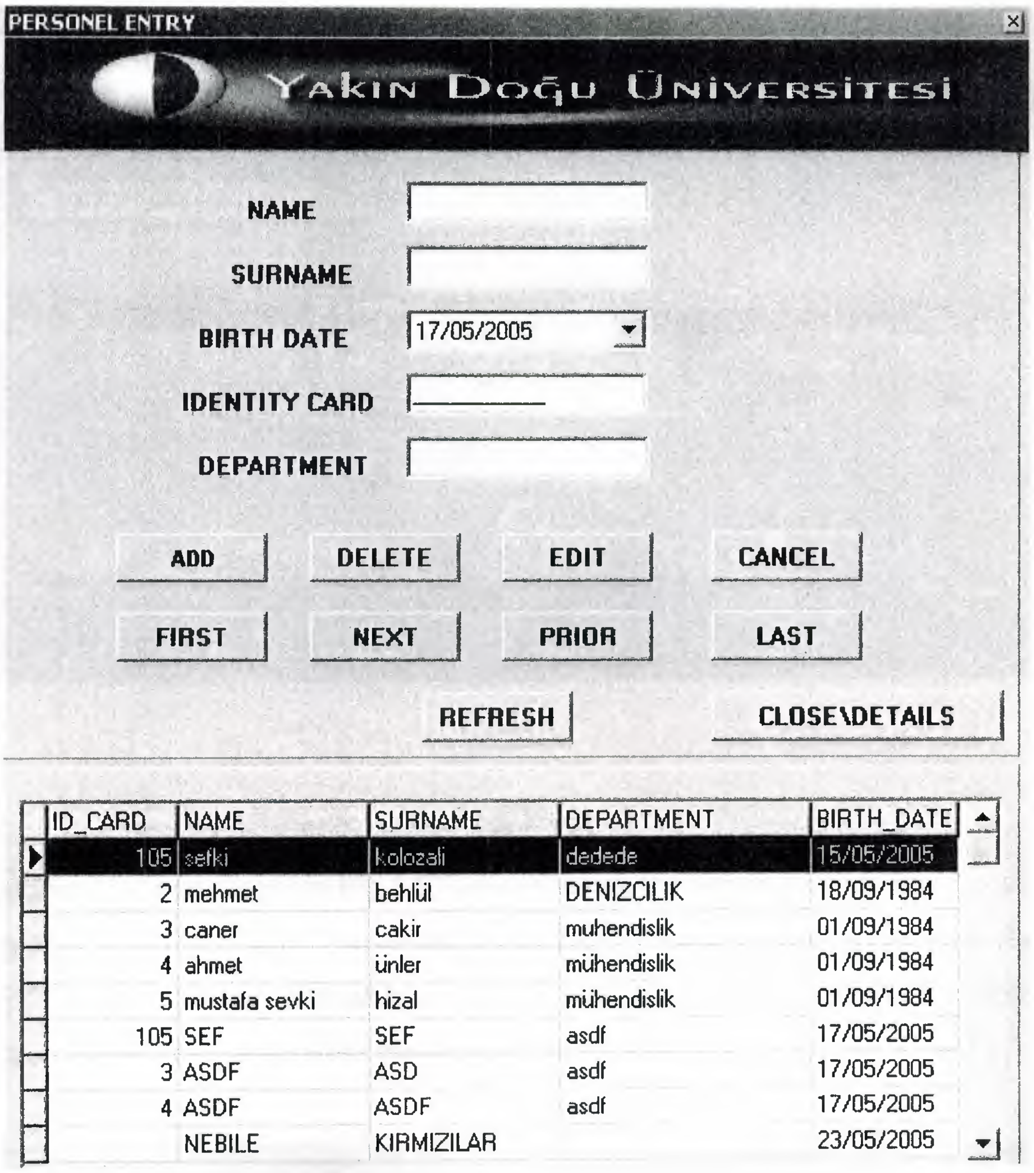
Fig 5.49

In the personal entry's part we can add ,delete and edit personals. For see the details we have to click details button in the personal entry's page. Its shown in Fig 5.50.when you click details button to see details it will opens details and button will change its name to close /detail for the close detail you will click again same button its shown in Fig 5.51



NAME
SURNAME
BIRTH DATE 17/05/2005
IDENTITY CARD
DEPARTMENT

Fig 5.50



NAME	
SURNAME	
BIRTH DATE	17/05/2005
IDENTITY CARD	
DEPARTMENT	

ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
105	sefki	kolozali	dedede	15/05/2005
2	mehmet	behlül	DENIZCILIK	18/09/1984
3	caner	cakir	muhendislik	01/09/1984
4	ahmet	ünler	mühendislik	01/09/1984
5	mustafa sevki	hizal	mühendislik	01/09/1984
105	SEF	SEF	asdf	17/05/2005
3	ASDF	ASD	asdf	17/05/2005
4	ASDF	ASDF	asdf	17/05/2005
	NEBILE	KIRMIZILAR		23/05/2005

Fig 5.51

The personal list part as you can see it Fig 5.52 when you click it, it will open new page its shown in Fig 5.53.

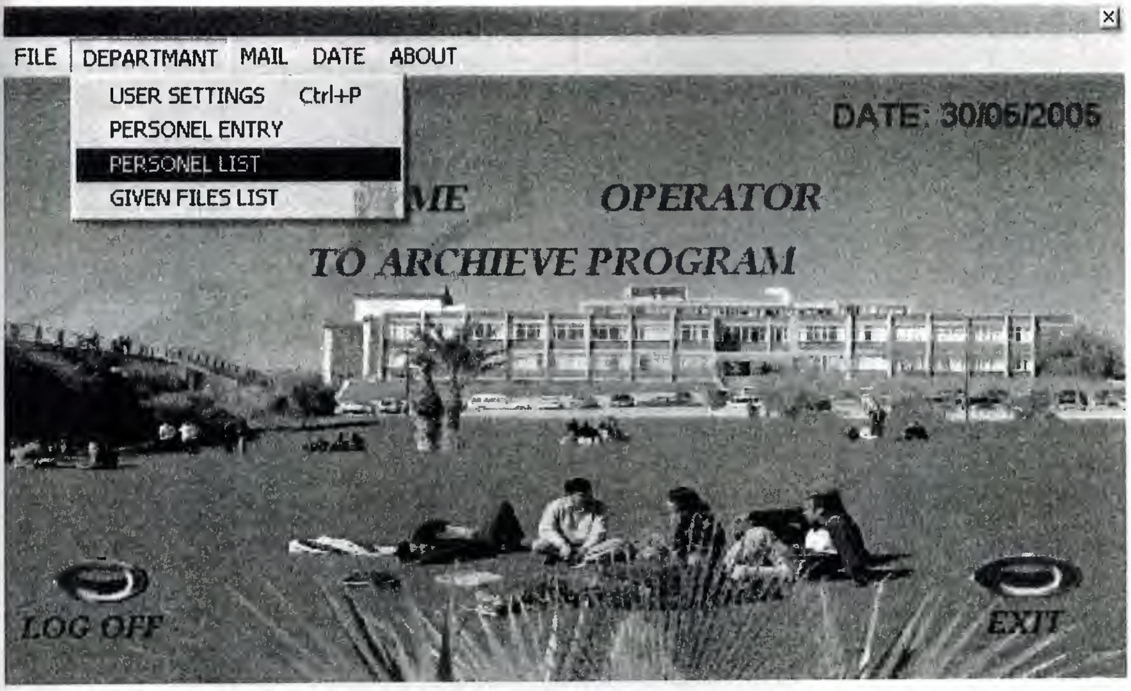


Fig 5.52

In personal list part shown in Fig 5.53 we can search the personal by id , name, surname , department and birth date .For see the list we have to click list option button as shown in Fig 5.54 and when we clicked it, it change its name to close/list and we can close the list from this button. If we want see all personal we have to click show all button.

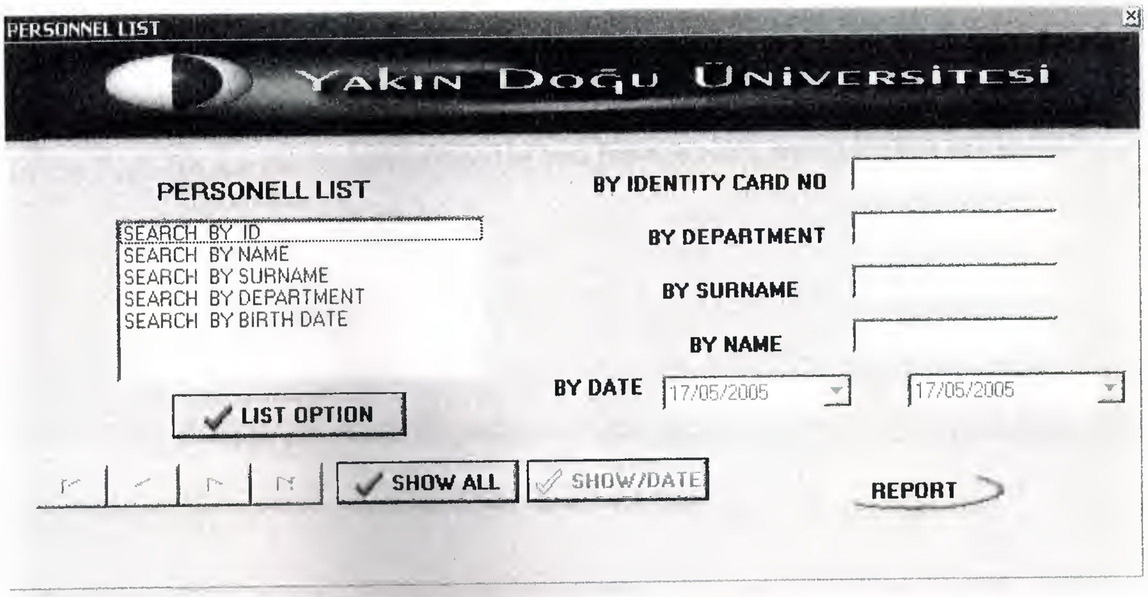
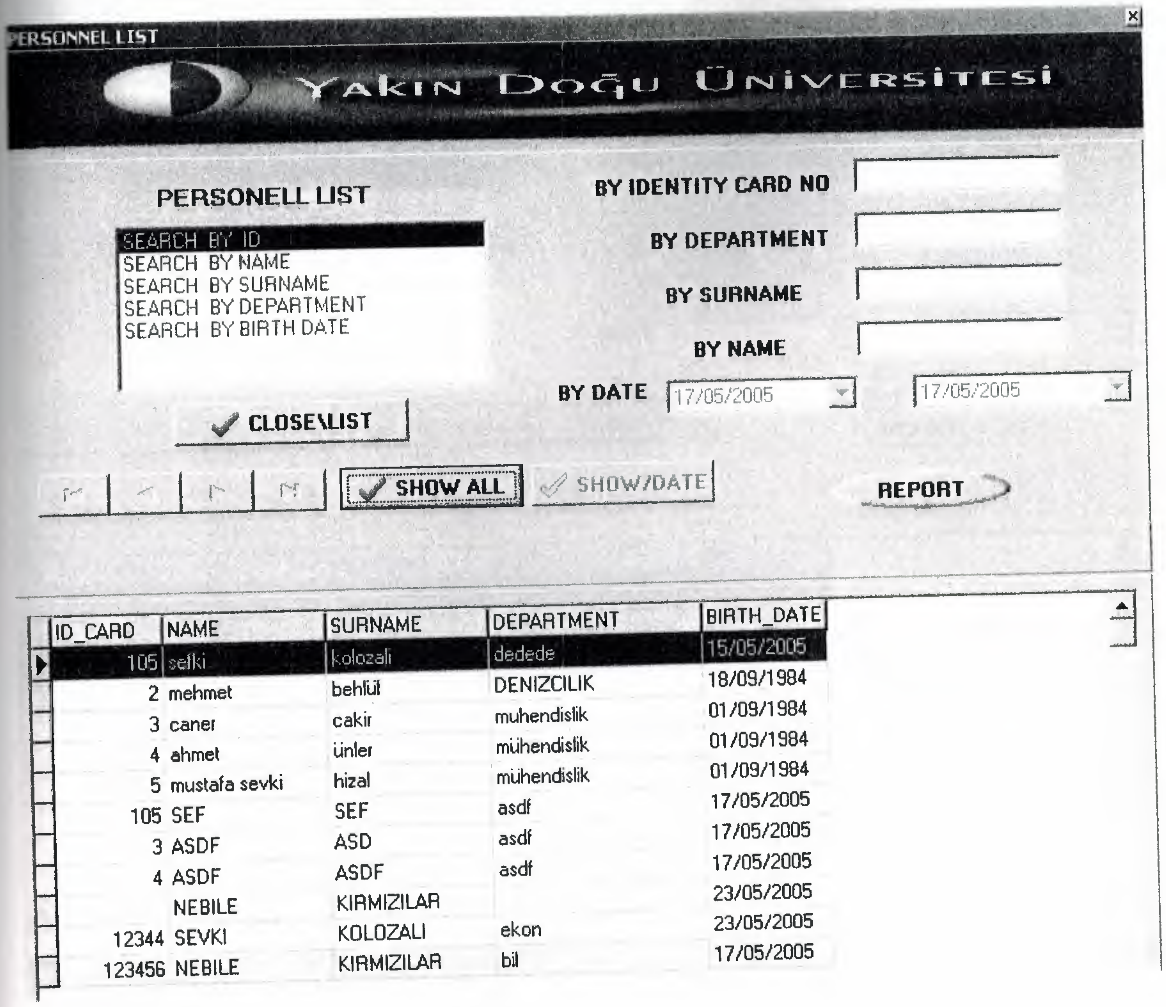


Fig 5.53

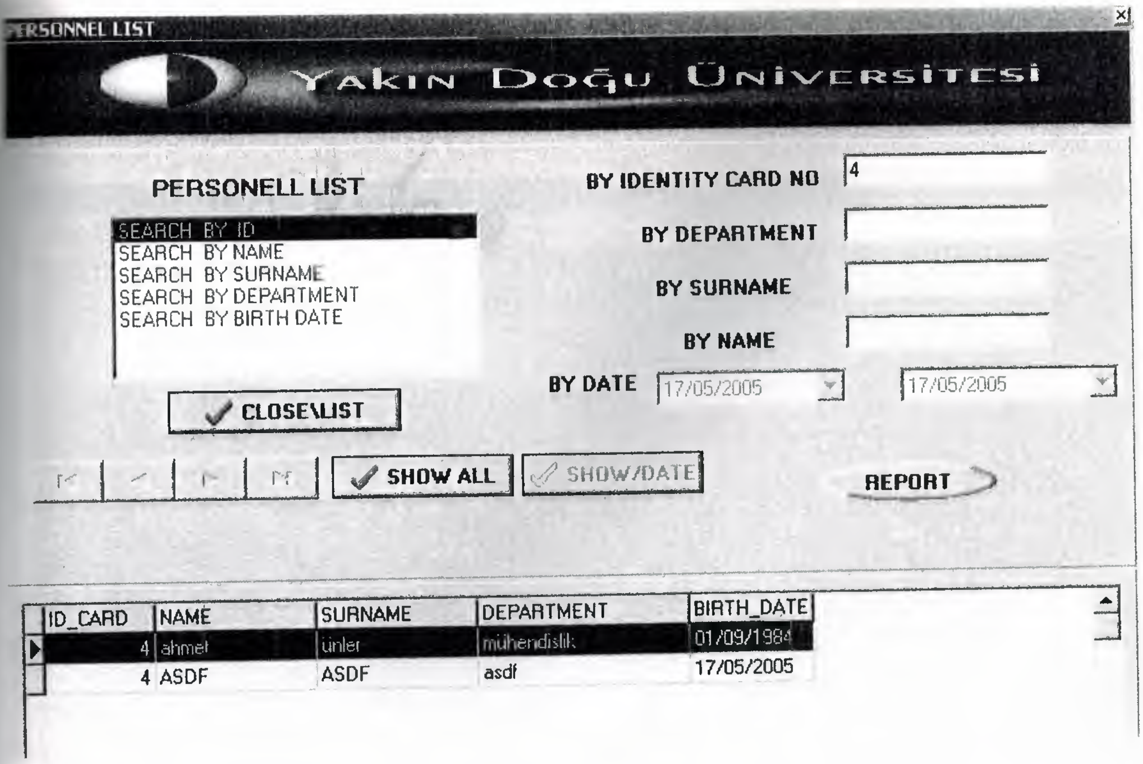


BY DATE 17/05/2005 17/05/2005

ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
105	sefki	kolozali	dedede	15/05/2005
2	mehmet	behlül	DENIZCILIK	18/09/1984
3	caner	cakir	muhendislik	01/09/1984
4	ahmet	ünler	mühendislik	01/09/1984
5	mustafa sevki	hizal	mühendislik	01/09/1984
105	SEF	SEF	asdf	17/05/2005
3	ASDF	ASD	asdf	17/05/2005
4	ASDF	ASDF	asdf	17/05/2005
	NEBILE	KIRMIZILAR		23/05/2005
12344	SEVKI	KOLOZALI	ekon	23/05/2005
123456	NEBILE	KIRMIZILAR	bil	17/05/2005

Fig 5.54

For make any search in this part we have to chose one part and make search its shown in Fig 5.55 .for see the reports of this list you have to click report button in Fig 5.55



PERSONELL LIST

- SEARCH BY ID
- SEARCH BY NAME
- SEARCH BY SURNAME
- SEARCH BY DEPARTMENT
- SEARCH BY BIRTH DATE

BY IDENTITY CARD NO 4

BY DEPARTMENT

BY SURNAME

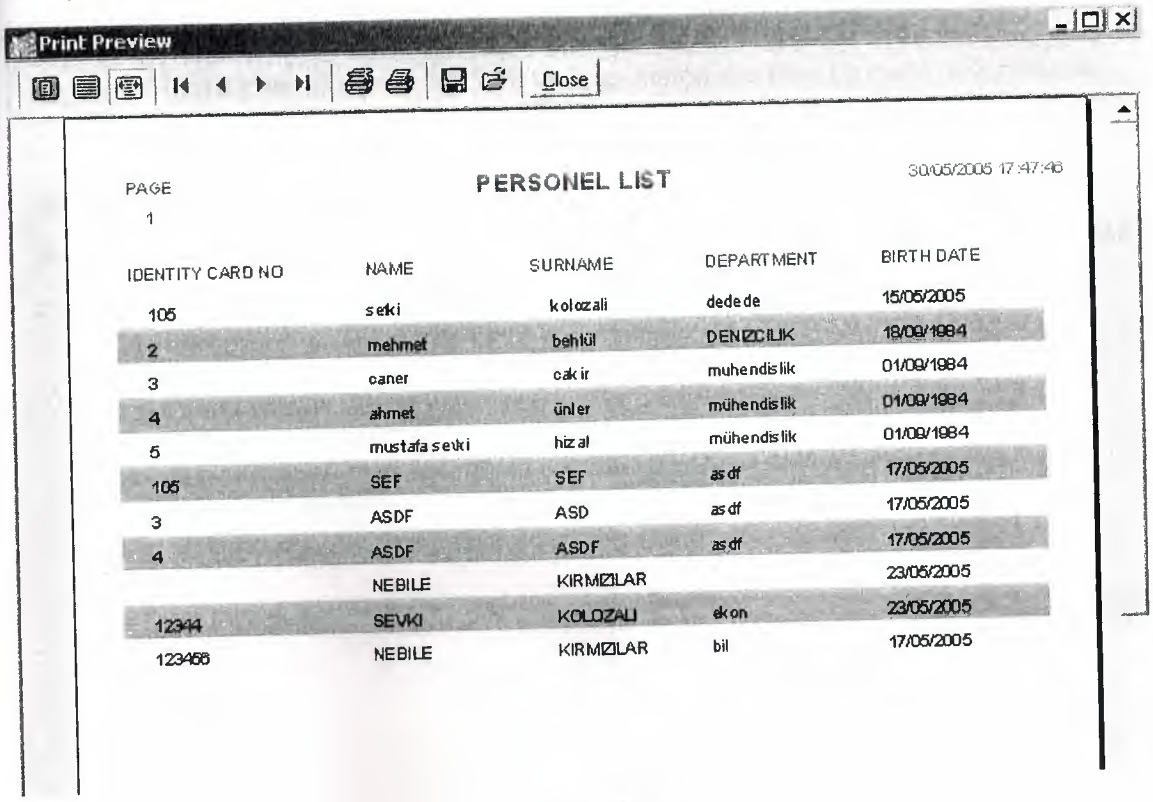
BY NAME

BY DATE 17/05/2005 17/05/2005

ID_CARD	NAME	SURNAME	DEPARTMENT	BIRTH_DATE
4	ahmet	ünler	mühendislik	01/09/1984
4	ASDF	ASDF	asdf	17/05/2005

Fig 5.55

And you can see your reports as shown in Fig 5.56



PAGE 1

PERSONEL LIST

30/05/2005 17:47:46

IDENTITY CARD NO	NAME	SURNAME	DEPARTMENT	BIRTH DATE
105	seki	kolozali	dedede	15/05/2005
2	mehmet	behlül	DENIZCILIK	18/09/1984
3	caner	cakir	muhendislik	01/09/1984
4	ahmet	ünler	mühendislik	01/09/1984
5	mustafa sevki	hizal	mühendislik	01/09/1984
105	SEF	SEF	asdf	17/05/2005
3	ASDF	ASD	asdf	17/05/2005
4	ASDF	ASDF	asdf	17/05/2005
	NEBILE	KIRMIZILAR		23/05/2005
12344	SEVKI	KOLOZALI	ekon	23/05/2005
123456	NEBILE	KIRMIZILAR	bil	17/05/2005

Fig 5.56

Last part of the department menu is given file list its shown in Fig 5.57 when you click it.

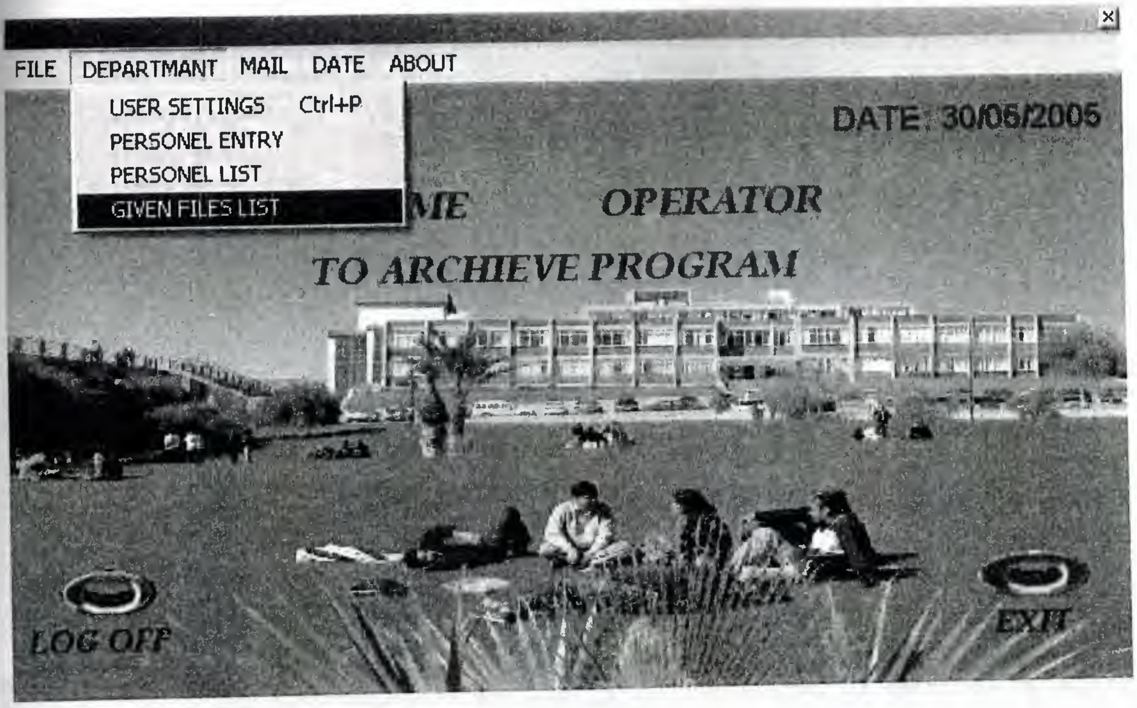


Fig 5.57

In given file list part shown in fig 5.58 we can search the files by code ,reference no , date , id , name and surname .For see the list we have to click list option button as shown in Fig 5.58 and when we clicked it ,it change its name to close/list and we can close the list from this button. If we want see all lists we have to click show all button. Shown in Fig 5.59.

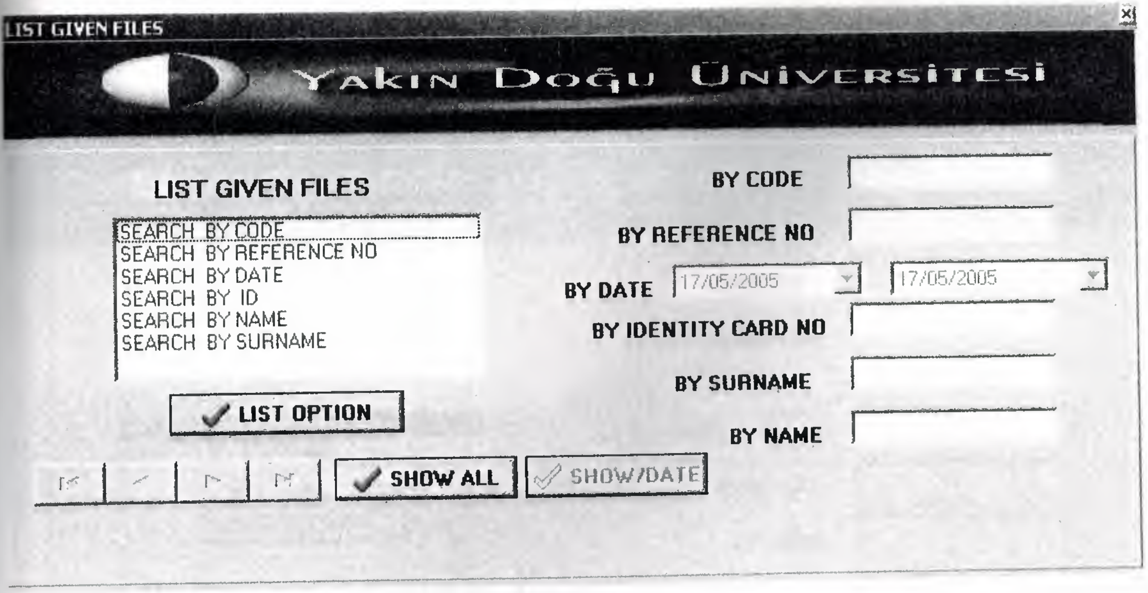
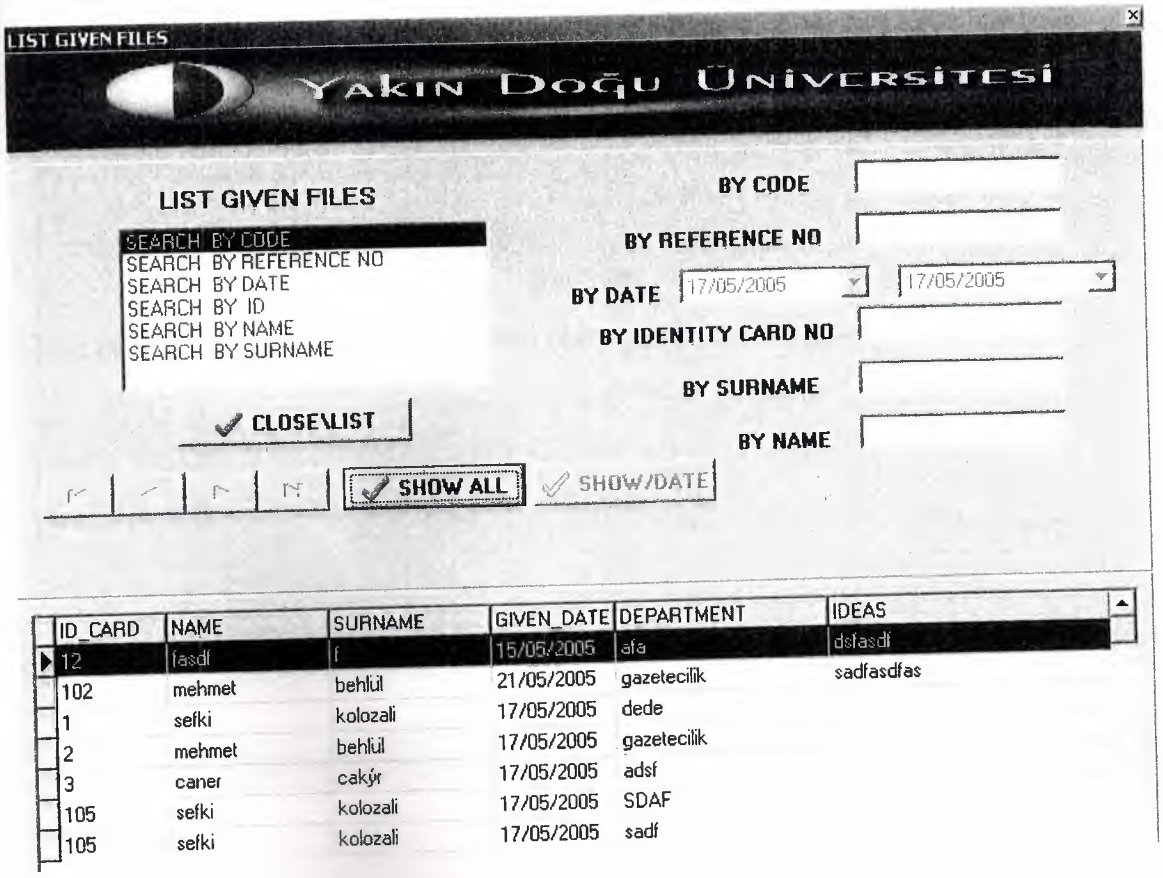


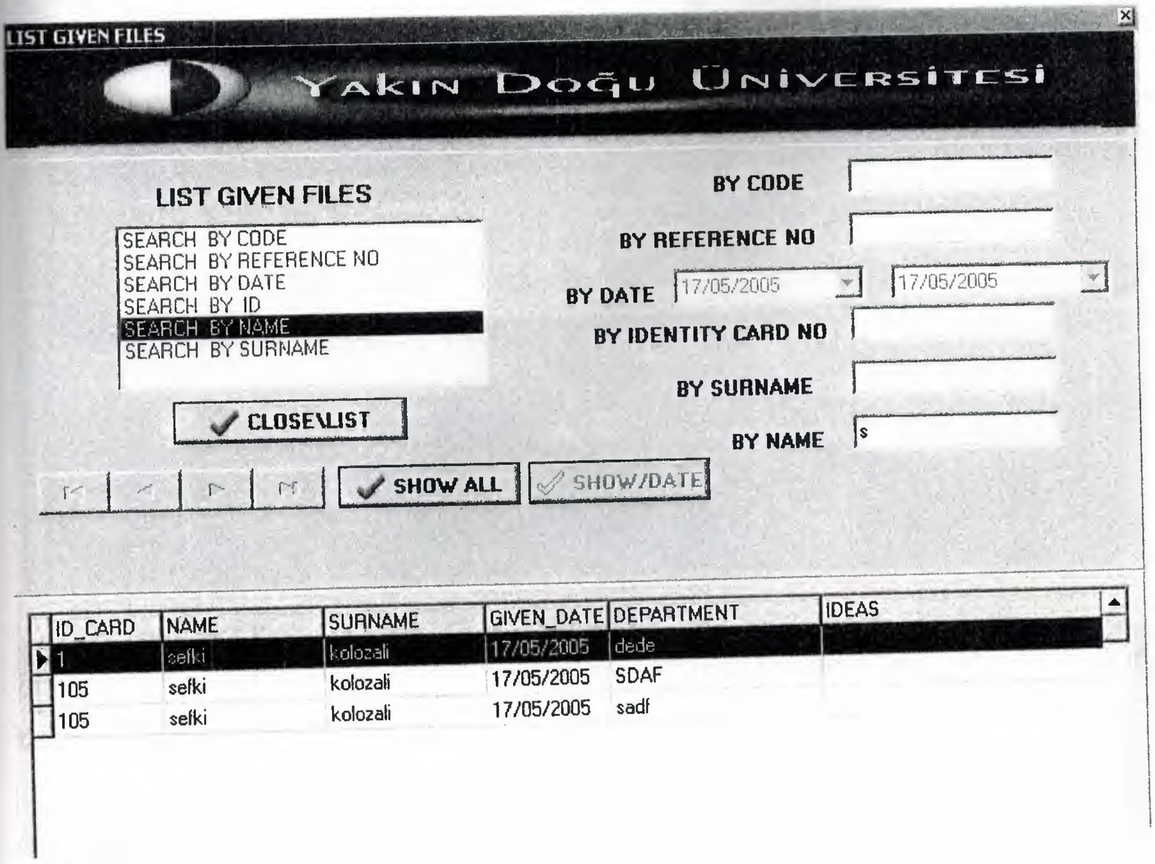
Fig 5.58



ID_CARD	NAME	SURNAME	GIVEN_DATE	DEPARTMENT	IDEAS
12	fasdf	f	15/05/2005	afa	dsfasdf
102	mehmet	behlül	21/05/2005	gazetecilik	sadfasdfas
1	sefki	kolozali	17/05/2005	dede	
2	mehmet	behlül	17/05/2005	gazetecilik	
3	caner	cakýr	17/05/2005	adsf	
105	sefki	kolozali	17/05/2005	SDAF	
105	sefki	kolozali	17/05/2005	sadf	

Fig 5.59

Which part as we like we will chose and write specification like shown in fig 5.60.



BY DATE: 17/05/2005 – 17/05/2005

BY NAME: s

ID_CARD	NAME	SURNAME	GIVEN_DATE	DEPARTMENT	IDEAS
1	sefki	kolozali	17/05/2005	dede	
105	sefki	kolozali	17/05/2005	SDAF	
105	sefki	kolozali	17/05/2005	sadf	

Fig 5.60

Third part of the our program is the mail menu as shown in Fig 5.61.

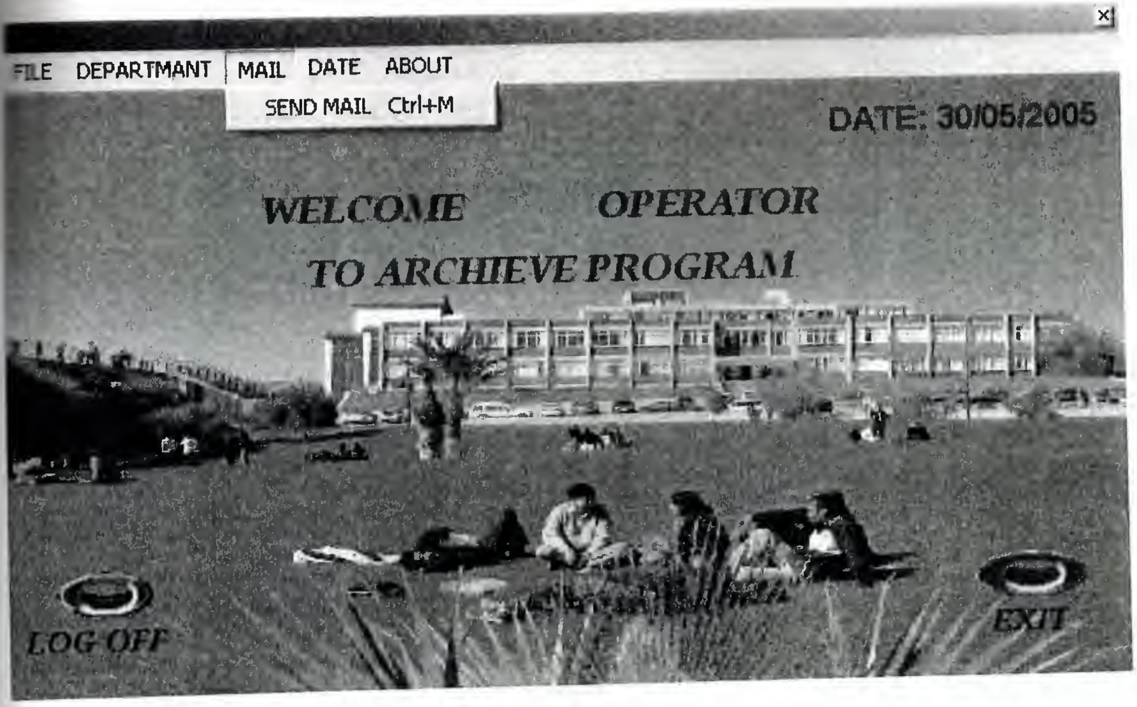


Fig 5.61

In our program without use outlook express we sent mail by using the send mail parts in the mail menus shown in Fig 5.62.

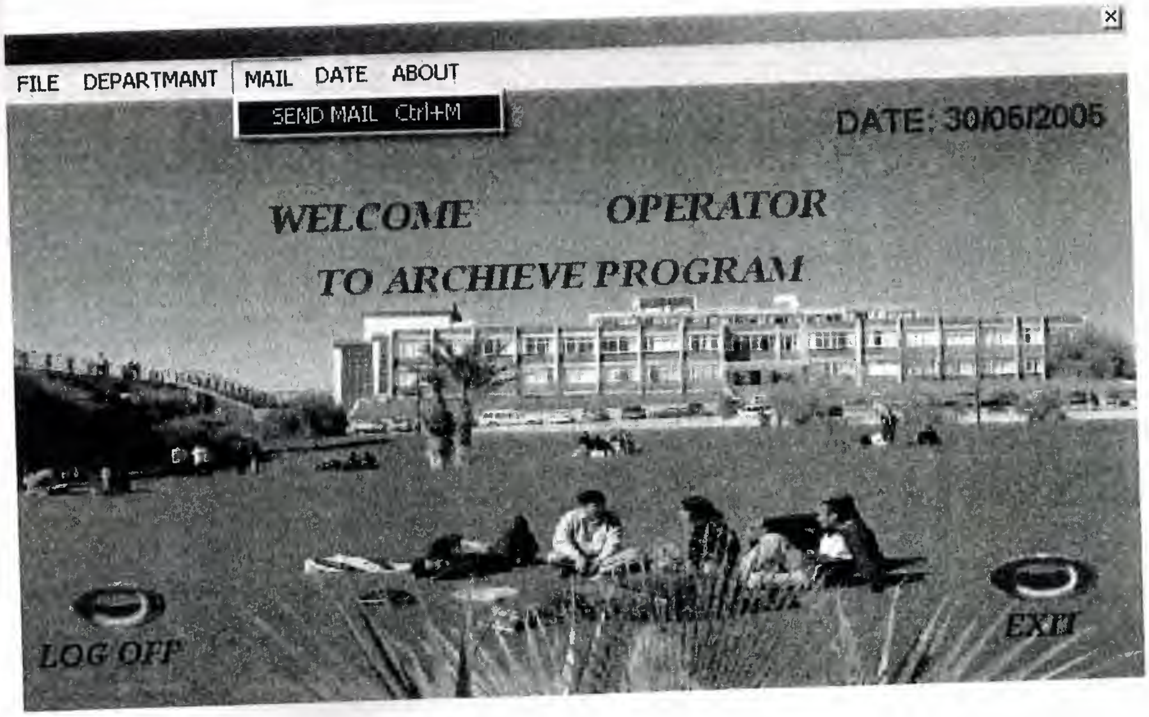
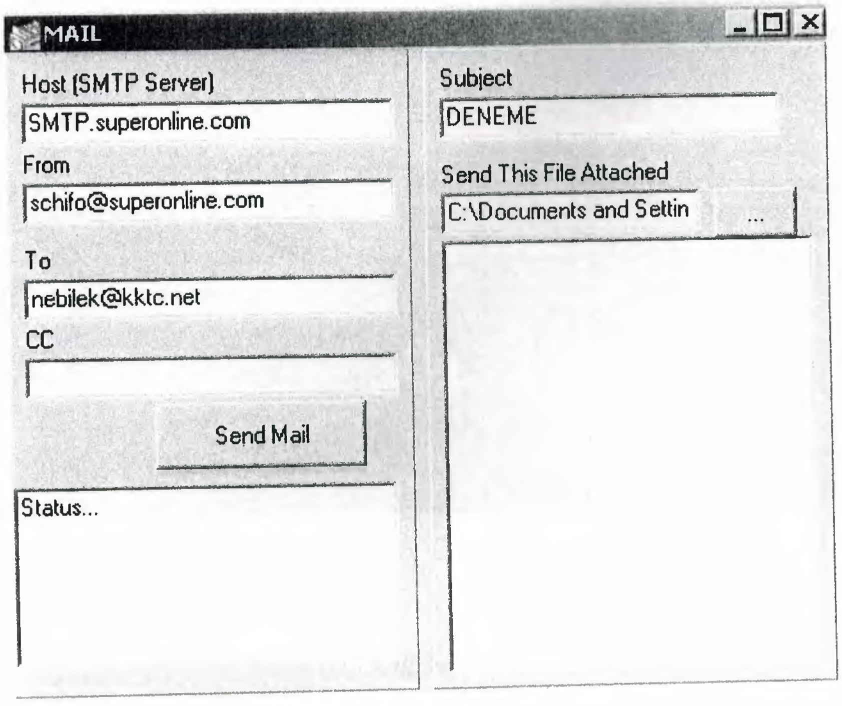


Fig 5.62

When we click the sent mail parts Fig 5.63 will open. With use this part we will send mail.



Host (SMTP Server)
SMTP.superonline.com
From
schifo@superonline.com
To
nebilek@kktc.net
CC

Status...

Subject
DENEME
Send This File Attached
C:\Documents and Settin

Fig 5.63

Firstly we have to write host its shown in Fig 5.63 it then click send mail button and send your mail.

The fourth menu in our program is the date menu's. In date's menu we have two part calendar and time parts as you see in Fig 5.64.

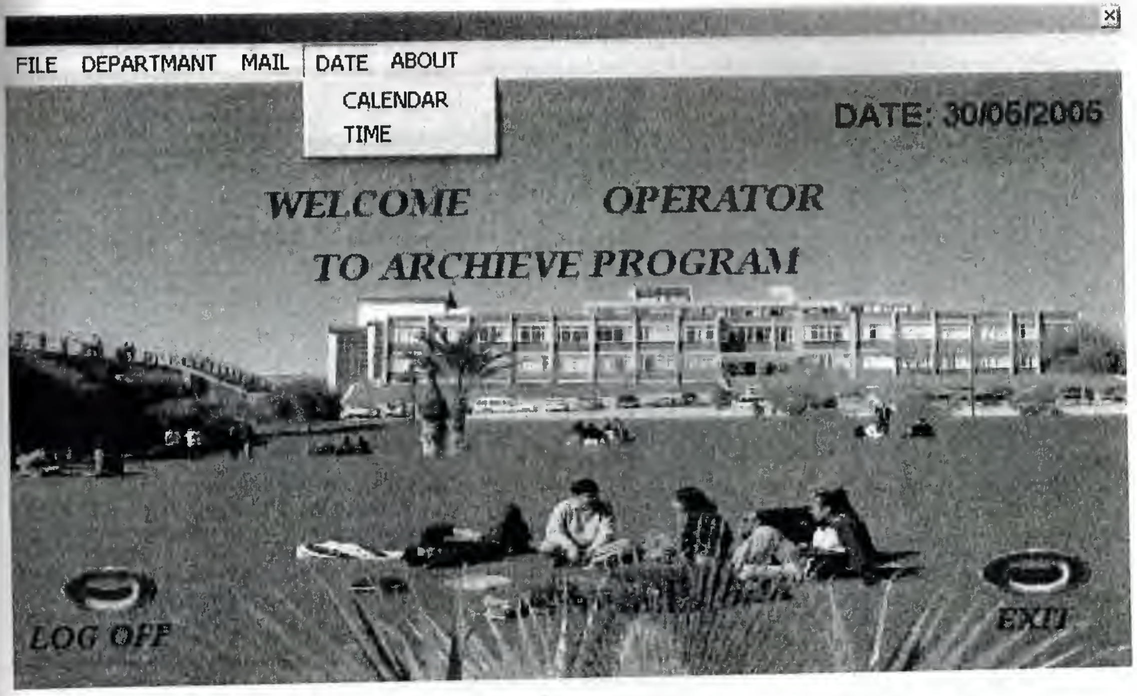


Fig 5.64

When you click the calendar has shown in fig 5.65 its open the calendar shown in Fig 5.66.

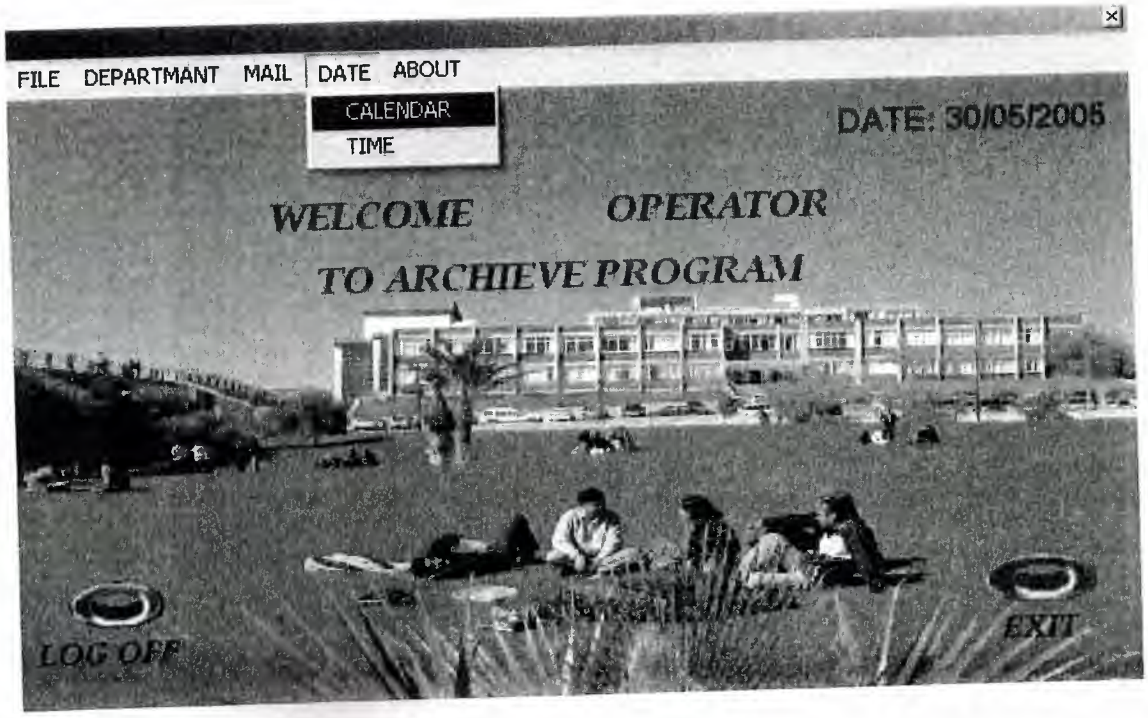
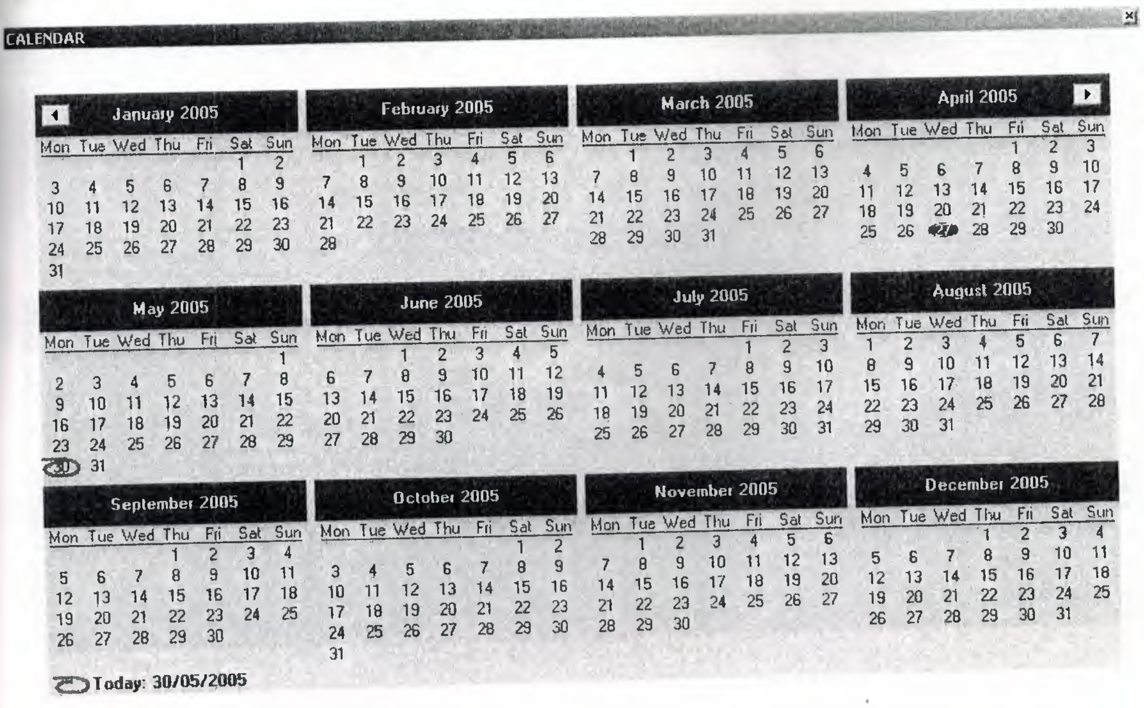


Fig 5.65



January 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

February 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

March 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

April 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

May 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

June 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

July 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

August 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

September 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

October 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

November 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

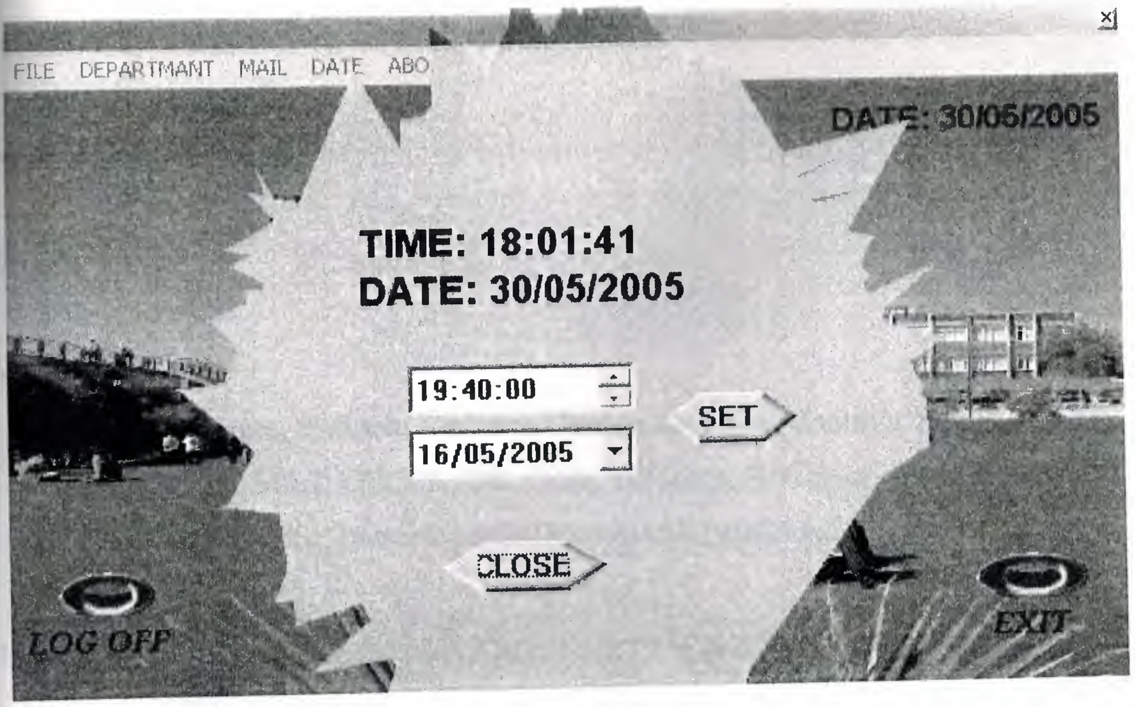
December 2005

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Today: 30/05/2005

Fig 5.66

If we want to set system time we have to click time in date menus shown in Fig 5.67



TIME: 18:01:41
DATE: 30/05/2005

19:40:00
16/05/2005

Fig 5.67

The last menu is about.In the about menu we have some information about how program working and about us.

APPENDIX

```
unit NEW;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Mask, DBCtrls, DB, Grids, DBGrids, DBTables, ExtCtrls,
  ComCtrls,FileCtrl, jpeg, Buttons, janTriDownButton, janRoundButton;

type
  Tform2 = class(TForm)
    DataSource1: TDataSource;
    Panel1: TPanel;
    Panel2: TPanel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    BitBtn5: TBitBtn;
    BitBtn6: TBitBtn;
    BitBtn7: TBitBtn;
    BitBtn8: TBitBtn;
    BitBtn9: TBitBtn;
    Edit1: TEdit;
    Edit2: TEdit;
    DateTimePicker1: TDateTimePicker;
```

```
Edit4: TEdit;
BitBtn10: TBitBtn;
Edit5: TEdit;
DBGrid1: TDBGrid;
Panel3: TPanel;
Image1: TImage;
Panel4: TPanel;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
DBEdit5: TDBEdit;
DBEdit6: TDBEdit;
janRoundButton1: TjanRoundButton;
DBEdit7: TDBEdit;
janRoundButton2: TjanRoundButton;
procedure FormCreate(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure Edit4Change(Sender: TObject);
procedure Edit5Change(Sender: TObject);
procedure Edit1KeyPress(Sender: TObject; var Key: Char);
procedure Edit2KeyPress(Sender: TObject; var Key: Char);
procedure Edit4KeyPress(Sender: TObject; var Key: Char);
procedure Edit5KeyPress(Sender: TObject; var Key: Char);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn5Click(Sender: TObject);
procedure BitBtn6Click(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);
procedure BitBtn9Click(Sender: TObject);
```

```
  procedure BitBtn8Click(Sender: TObject);
  procedure BitBtn10Click(Sender: TObject);
  procedure MaskEdit1KeyPress(Sender: TObject; var Key: Char);
  procedure FormActivate(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure BitBtn14Click(Sender: TObject);
  procedure BitBtn3Click(Sender: TObject);
  procedure BitBtn7Click(Sender: TObject);
  procedure janRoundButton1Click(Sender: TObject);
  procedure janRoundButton2Click(Sender: TObject);
  procedure DateTimePicker1KeyPress(Sender: TObject; var Key: Char);
  procedure Panel1Click(Sender: TObject);

 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
  form2: Tform2;

implementation

uses DataModule, CREATEFILE, GIRIS,EDIT, explorer, GIVE_FILE;

{$R *.dfm}

procedure Tform2.FormCreate(Sender: TObject);
begin
DateTimePicker1.date:=date;
form2.Width:=666;
form2.Height:=416;
```

```
edit1.Clear;
edit2.Clear;
edit4.Clear;
edit5.Clear;
end;

procedure Tform2.Button7Click(Sender: TObject);
begin
data.dosya1.last;
end;

procedure Tform2.Edit1Change(Sender: TObject);
begin
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit2Change(Sender: TObject);
begin
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit3Change(Sender: TObject);
begin
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit4Change(Sender: TObject);
begin
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit5Change(Sender: TObject);
begin
```

```
font.Style:=[fsbold] ;
end;

procedure Tform2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
  if (SelStart = 0) or
    (Text[SelStart] = ' ') then
      if Key in ['a'..'z'] then
        Key := UpCase(Key);
    if key=#13 then edit2.setfocus;
if key=#13 then edit2.setfocus;
end;

procedure Tform2.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
  if (SelStart = 0) or
    (Text[SelStart] = ' ') then
      if Key in ['a'..'z'] then
        Key := UpCase(Key);
        if key=#13 then DateTimePicker1.setfocus;
if key=#13 then DateTimePicker1.setfocus;
end;

procedure Tform2.Edit4KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
  if (SelStart = 0) or
    (Text[SelStart] = ' ') then
      if Key in ['a'..'z'] then
        Key := UpCase(Key);
  if key=#13 then edit5.setfocus;
```

```
end;

procedure Tform2.Edit5KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
  if (SelStart = 0) or
    (Text[SelStart] = ' ') then
      if Key in ['a'..'z'] then
        Key := UpCase(Key);
  if key=#13 then BitBtn1.setfocus;
end;

procedure Tform2.BitBtn1Click(Sender: TObject);
var
btn:integer;
begin
if (edit1.Text='') or (edit2.Text='') or (edit4.Text='') or (edit5.Text='') then
 ShowMessage('PLEASE ENTER REPORT IDENTIFICATIONS')
 else
 begin
 DATA.dosya1.Filter:='CODE='+quotedstr(edit1.Text) + 'and
REF_NO='+quotedstr(edit2.Text);
 DATA.dosya1.Filtered:=true;

 if (data.dosya1.FieldValues['CODE']<>edit1.Text) and
 (data.dosya1.FieldValues['REF_NO']<>edit2.Text) then
  begin
  btn:=MessageDlg('DO YOU WANT TO INSERT ANY FILE TO
 PROGRAM?',mtWarning,[mbYes,mbNo,mbCancel],0);
   if btn=6 then
    begin
     forcedirectories('c:\DATAS\RECEIVED FILES\'+ edit1.text + '\' +edit2.text);
     form4.ShowModal;
 end;
  end
 else
    BEGIN
    showmessage('This file is already exits');
    END;
if btn=7 then
begin
data.dosya1.open;
data.dosya1.append;
data.dosya1.fieldvalues['CODE']:=edit1.text;
data.dosya1.fieldvalues['REF_NO']:=edit2.text;
data.dosya1.fieldvalues['DATE']:=DateTimePicker1.DateTime;
data.dosya1.fieldvalues['TOPIC']:=edit4.text;
data.dosya1.fieldvalues['NAME']:=edit5.text;
data.dosya1.post;
end;
end;
data.dosya1.Filtered:=false;
edit1.clear;
edit2.clear;
edit4.clear;
edit5.clear;
edit1.setfocus;
end;

procedure Tform2.BitBtn2Click(Sender: TObject);
var
btn:integer;
begin
btn:=MessageDlg('DO YOU WANT TO
DELETE?',mtConfirmation,[mbYes,mbNo],0);
if btn= 6 then
```

```
 begin
  data.dosya1.Delete;
  dbgrid1.Update;
 end;
end;
procedure Tform2.BitBtn5Click(Sender: TObject);
begin
data.dosya1.first;
end;

procedure Tform2.BitBtn6Click(Sender: TObject);
begin
data.dosya1.prior;
end;

procedure Tform2.BitBtn4Click(Sender: TObject);
begin
data.dosya1.next;
end;

procedure Tform2.BitBtn9Click(Sender: TObject);
begin
data.dosya1.cancel;
end;

procedure Tform2.BitBtn8Click(Sender: TObject);
var
 btn:integer;
begin
  btn:=MessageDlg('DO YOU WANT TO UPDATE
IT',mtConfirmation,[mbYes,mbNo],0);
   if btn= 6 then
    begin
   FORM15.Show;
  end;
end;

procedure Tform2.BitBtn10Click(Sender: TObject);
begin
data.dosya1.refresh;
end;

procedure Tform2.MaskEdit1KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then edit4.setfocus;
end;

procedure Tform2.FormActivate(Sender: TObject);
begin
data.dosya1.Open;
end;

procedure Tform2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
data.dosya1.Close;
end;

procedure Tform2.BitBtn14Click(Sender: TObject);
var
 btn:integer;
begin
  btn:=MessageDlg('DO YOU WANT TO UPDATE
IT?',mtConfirmation,[mbYes,mbNo],0);
   if btn= 6 then
    begin
  FORM15.Show;
  end;
end;

procedure Tform2.BitBtn3Click(Sender: TObject);
begin
 if (form2.Width=666) and (form2.Height=416) then
   begin
     form2.Width:=666;
     form2.Height:=636;
     panel2.Visible:=true;
     DBGrid1.Visible:=true;
     BitBtn3.Caption:='CLOSE\DETAILS';
    end
  else
  begin
   form2.Width:=666;
   form2.Height:=416;
   panel2.Visible:=false;
   DBGrid1.Visible:=false;
   BitBtn3.Caption:='DETAILS';
   end;
  end;

  procedure Tform2.BitBtn7Click(Sender: TObject);
  begin
  data.dosya1.Last;
  end;

  procedure Tform2.janRoundButton1Click(Sender: TObject);
  begin
   if DBEdit7.Text='' then
 ShowMessage('THERE IS NO INSERTED FILE')
 else
 begin
  form17.shelllistview1.root:= ExtractFilePath(dbedit7.text);
  form17.Show;
end;
end;

procedure Tform2.janRoundButton2Click(Sender: TObject);
begin
GIVE_FILES.show;
end;

procedure Tform2.DateTimePicker1KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then edit4.setfocus;
end;
```

unit CREATEFILE;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Spin, DBCtrls, ExtDlgs, ExtCtrls, janRoundButton;

type
  TForm4 = class(TForm)
    DBImage1: TDBImage;
    SpinEdit1: TSpinEdit;
    OpenPictureDialog1: TOpenPictureDialog;
    RadioButton2: TRadioButton;
    RadioButton1: TRadioButton;
  OpenDialog1: TOpenDialog;
  SELECT: TRadioGroup;
  janRoundButton1: TjanRoundButton;
  procedure Button1Click(Sender: TObject);
  procedure DBImage1DblClick(Sender: TObject);
  procedure RadioButton2Click(Sender: TObject);
  procedure RadioButton1Click(Sender: TObject);
  procedure janRoundButton1Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form4: TForm4;

implementation

uses NEW, Datamodule;

{$R *.dfm}

procedure TForm4.Button1Click(Sender: TObject);
begin
if not fileexists('c:\DATAS\RECEIVED FILES\'+form2.edit1.text+'\'+
form2.edit2.Text+ '\' +inttostr(SpinEdit1.value)+'.BMP') then
    begin
    DBImage1.Picture.SaveToFile('c:\DATAS\RECEIVED FILES\'+form2.edit1.text +
'\' + form2.edit2.Text+'\' +inttostr(SpinEdit1.value)+'.BMP');
    SpinEdit1.Value := spinedit1.Value + 1;
    DATA.dosya1.Open;
```

```
   data.dosya1.Append;
data.dosya1.fieldvalues['CODE']:=form2.edit1.text;
data.dosya1.fieldvalues['REF_NO']:=form2.edit2.text;
data.dosya1.fieldvalues['DATE']:=form2.DateTimePicker1.DateTime;
data.dosya1.fieldvalues['TOPIC']:=form2.edit4.text;
data.dosya1.fieldvalues['NAME']:=form2.edit5.text;
   data.dosya1.FieldValues['INDEX']:='c:\DATAS\RECEIVED FILES\'+
FORM2.edit1.text + '\' +FORM2.edit2.text;
   data.dosya1.Post;
   end
else
  begin
        if messagedlg('Bu sayfa mevcut. Üstüne kaydedeyim mi
',mtconfirmation,[mbyes,mbno],0) = mryes then
             DBImage1.Picture.SaveToFile('c:\DATAS\RECEIVED
FILES\'+form2.edit1.text+ '\' +form2.edit2.text + '\' +inttostr(SPinEdit1.value)+
'.BMP');
    end;
    end;

procedure TForm4.DBImage1DblClick(Sender: TObject);
begin
openpicturedialog1.title:='resim dosyasý seçiniz';
openpicturedialog1.filter:='resim dosyalarý|*.BMP';
openpicturedialog1.filterindex:=0;
if(openpicturedialog1.execute) then
 begin
 data.dosya1.Open;
   data.dosya1.edit;
   dbimage1.Picture.LoadFromFile(openpicturedialog1.filename);
   data.dosya1.Post;
  end
 end;
```

```
procedure TForm4.RadioButton2Click(Sender: TObject);
begin
form4.Height:=600;
DBimage1.Height:=500;
form4.Width:=700;
dbimage1.Visible:=true;
janRoundButton1.Visible:=true;
spinedit1.Visible:=true;
end;

procedure TForm4.RadioButton1Click(Sender: TObject);
var address:string;
  NewFileName: string;
  Msg: string;
  NewFile: TFileStream;
  OldFile: TFileStream;
begin
form4.Height:=213;
dbimage1.Height:=0;
dbimage1.Visible:=false;
form4.Width:=181;
janRoundButton1.Visible:=false;
spinedit1.Visible:=false;
if opendialog1.Execute then
       begin
       address:=opendialog1.Filename;
  NewFileName := 'c:\DATAS\RECEIVED
FILES\'+form2.edit1.text+'\'+form2.Edit2.Text+ '\'+ExtractFileName(address);
  Msg := Format('Copy %s to %s?', [address, NewFileName]);
  if MessageDlg(Msg, mtCustom, mbOKCancel, 0) = mrOK then
  begin
    OldFile := TFileStream.Create(ExtractFileName(address), fmOpenRead or
fmShareDenyWrite);
```

```
      NewFile := TFileStream.Create(NewFileName, fmCreate or fmShareDenyWrite);
        NewFile.CopyFrom(OldFile, OldFile.Size);
        data.dosya1.Open;
        data.dosya1.append;
        data.dosya1.FieldValues['INDEX']:='c:\DATAS\RECEIVED
FILES\'+form2.edit1.text+'\'+form2.Edit2.Text+ '\'+ExtractFileName(address);
        data.dosya1.fieldvalues['CODE']:=form2.edit1.text;
        data.dosya1.fieldvalues['REF_NO']:=form2.edit2.text;
        data.dosya1.fieldvalues['DATE']:=form2.DateTimePicker1.DateTime;
        data.dosya1.fieldvalues['TOPIC']:=form2.edit4.text;
        data.dosya1.fieldvalues['NAME']:=form2.edit5.text;
        data.dosya1.post;
        showmessage('Document is successfully saved');
    end;
         end;
         form4.Close;
end;

procedure TForm4.janRoundButton1Click(Sender: TObject);
begin
if not fileexists('c:\DATAS'+form2.edit1.text+'\'+ form2.edit2.Text+ '\'
+inttostr(SpinEdit1.value)+'.BMP') then
     begin
     DBImage1.Picture.SaveToFile('c:\DATAS\RECEIVED FILES\'+form2.edit1.text +
'\' + form2.edit2.Text+'\' +inttostr(SpinEdit1.value)+'.BMP');
     SpinEdit1.Value := spinedit1.Value + 1;
     DATA.dosya1.Open;
     data.dosya1.Append;
     data.dosya1.fieldvalues['CODE']:=form2.edit1.text;
     data.dosya1.fieldvalues['REF_NO']:=form2.edit2.text;
     data.dosya1.fieldvalues['DATE']:=form2.DateTimePicker1.DateTime;
     data.dosya1.fieldvalues['TOPIC']:=form2.edit4.text;
     data.dosya1.fieldvalues['NAME']:=form2.edit5.text;
    data.dosya1.FieldValues['INDEX']:='c:\DATAS\RECEIVED FILES'+'\'+
FORM2.edit1.text + '\' +FORM2.edit2.text;
    data.dosya1.Post;
    showmessage('it is saved');
    end
else
  begin
        if messagedlg('this file is exist. do you want to record on to
it',mtconfirmation,[mbyes,mbno],0) = mryes then
            DBImage1.Picture.SaveToFile('c:\DATAS\RECEIVED
FILES\'+form2.edit1.text+ '\' +form2.edit2.text + '\' +inttostr(SPinEdit1.value)+
'.BMP');
     end;
     end;
procedure TForm4.FormCreate(Sender: TObject);
begin
end;
end.
```

unit LIST_RECEIVED_FILES;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Mask, Grids, DBGrids, DB, DBTables, ComCtrls, jpeg,
  ExtCtrls, janRoundButton, DBCtrls, Buttons;

type
  TForm7 = class(TForm)
    DataSource1: TDataSource;
    Query1: TQuery;
    Query1CODE: TStringField;
```

```
Query1REF_NO: TStringField;
Query1DATE: TDateField;
Query1TOPIC: TStringField;
Query1NAME: TStringField;
Query1INDEX: TStringField;
Panel1: TPanel;
ListBox1: TListBox;
Label1: TLabel;
BitBtn1: TBitBtn;
Panel2: TPanel;
Edit1: TEdit;
Panel3: TPanel;
Image1: TImage;
Edit2: TEdit;
Edit3: TEdit;
DBGrid1: TDBGrid;
Label2: TLabel;
DateTimePicker1: TDateTimePicker;
DateTimePicker2: TDateTimePicker;
Edit4: TEdit;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
janRoundButton1: TjanRoundButton;
DBEdit1: TDBEdit;
DBNavigator1: TDBNavigator;
janRoundButton2: TjanRoundButton;
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure janRoundButton1Click(Sender: TObject);
procedure DBGrid2CellClick(Column: TColumn);
procedure FormActivate(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
procedure Edit4Change(Sender: TObject);
procedure Edit3DblClick(Sender: TObject);
procedure Edit4DblClick(Sender: TObject);
procedure Edit2DblClick(Sender: TObject);
procedure Edit1DblClick(Sender: TObject);
procedure Edit3KeyPress(Sender: TObject; var Key: Char);
procedure Edit2KeyPress(Sender: TObject; var Key: Char);
procedure Edit4KeyPress(Sender: TObject; var Key: Char);
procedure Edit1KeyPress(Sender: TObject; var Key: Char);
procedure janRoundButton2Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure DateTimePicker1KeyPress(Sender: TObject; var Key: Char);
procedure DateTimePicker2KeyPress(Sender: TObject; var Key: Char);
procedure Panel1Click(Sender: TObject);

private
  { Private declarations }
public
  { Public declarations }
end;

var
 Form7: TForm7;
 docaddress:string;

implementation
```

```
uses Datamodule,comobj, explorer, REPORTS;

{$R *.dfm}
procedure TForm7.Edit1Change(Sender: TObject);
begin
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('select * from dosya1');
Query1.SQL.Add('WHERE NAME Like' + QUOTEDSTR(Edit1.Text+'%'));
Query1.Open;
end;

procedure TForm7.Edit2Change(Sender: TObject);
begin
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('select * from dosya1');
Query1.SQL.Add('WHERE TOPIC Like' + QUOTEDSTR(Edit2.Text+'%'));
Query1.Open;
end;

procedure TForm7.Edit3Change(Sender: TObject);
begin
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('SELECT * FROM DOSYA1');

Query1.SQL.Add('WHERE CODE Like' + QUOTEDSTR(Edit3.Text+'%'));
Query1.Open;
end;
```

```
procedure TForm7.janRoundButton1Click(Sender: TObject);
begin
 if DBEdit1.Text='' then
  ShowMessage('THERE IS NO INSERTED FILE')
  else
  begin
   form17.shelllistview1.root:= ExtractFilePath(dbedit1.text);
   form17.Show;
end;
end;

procedure TForm7.DBGrid2CellClick(Column: TColumn);
begin
docaddress:= data.dosya1INDEX.asstring;
end;

procedure TForm7.FormActivate(Sender: TObject);
begin
DATA.dosya1.Open;
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('select * from dosya1');
Query1.Open;
Edit1.Clear;
Edit2.Clear;
Edit3.Clear;
Edit4.Clear;

 BitBtn2.Enabled:=False;
 DateTimePicker1.Enabled:=false;
 DateTimePicker2.Enabled:=false;
 Form7.Width:=668;
 Form7.Height:=324;
```

```
end;

procedure TForm7.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   Query1.Close;
   data.dosya1.close;
end;

procedure TForm7.BitBtn1Click(Sender: TObject);
begin
 if ListBox1.ItemIndex=0 THEN
  begin
   if (form7.Width=668) and (form7.Height=324) then
     begin
     form7.Width:=668;
     form7.Height:=622;
     panel2.Visible:=true;
     DBGrid1.Visible:=true;
     BitBtn1.Caption:='CLOSE\LIST';
     edit3.Enabled:=true;
     BitBtn1.Caption:='CLOSE\LIST';
     end
     else
  begin
   form7.Width:=668;
   form7.Height:=324;
   panel2.Visible:=false;
   DBGrid1.Visible:=false;
   BitBtn1.Caption:='LIST';
   BitBtn2.Enabled:=False;
   DateTimePicker1.Enabled:=false;
   DateTimePicker2.Enabled:=false;
  end;
  end;
if ListBox1.ItemIndex=1 THEN
  begin
   if (form7.Width=668) and (form7.Height=324) then
     begin
     form7.Width:=668;
     form7.Height:=622;
     panel2.Visible:=true;
     DBGrid1.Visible:=true;
     BitBtn1.Caption:='CLOSE\LIST';
     edit4.Enabled:=true;
     BitBtn1.Caption:='CLOSE\LIST';
     end
     else
  begin
   form7.Width:=668;
   form7.Height:=324;
   panel2.Visible:=false;
   DBGrid1.Visible:=false;
   BitBtn1.Caption:='LIST';

   BitBtn2.Enabled:=False;
   DateTimePicker1.Enabled:=false;
   DateTimePicker2.Enabled:=false;
  end;
   end;
 if ListBox1.ItemIndex=2 THEN
   begin
    if (form7.Width=668) and (form7.Height=324) then
      begin
      form7.Width:=668;
      form7.Height:=622;
      panel2.Visible:=true;
  DBGrid1.Visible:=true;
  BitBtn1.Caption:='CLOSE\LIST';
  BitBtn2.Enabled:=True;
  DateTimePicker1.Enabled:=true;
  DateTimePicker2.Enabled:=true;
  BitBtn1.Caption:='CLOSE\LIST';
  end
  else
begin
 form7.Width:=668;
 form7.Height:=324;
 panel2.Visible:=false;
 DBGrid1.Visible:=false;
 BitBtn1.Caption:='LIST';

 BitBtn2.Enabled:=False;
 DateTimePicker1.Enabled:=false;
 DateTimePicker2.Enabled:=false;
 end;
  end;
 if ListBox1.ItemIndex=3 THEN
  begin
   if (form7.Width=668) and (form7.Height=324) then
     begin
     form7.Width:=668;
     form7.Height:=622;
     panel2.Visible:=true;
     DBGrid1.Visible:=true;
     BitBtn1.Caption:='CLOSE\LIST';
     edit2.Enabled:=true;
     BitBtn1.Caption:='CLOSE\LIST';
     end
     else
```

```
begin
 form7.Width:=668;
 form7.Height:=324;
 panel2.Visible:=false;
 DBGrid1.Visible:=false;
 BitBtn1.Caption:='LIST';
end;
 end;
if ListBox1.ItemIndex=4 THEN
  begin
   if (form7.Width=668) and (form7.Height=324) then
     begin
     form7.Width:=668;
     form7.Height:=622;
     panel2.Visible:=true;
     DBGrid1.Visible:=true;
     BitBtn1.Caption:='CLOSE\LIST';
     edit1.Enabled:=true;
     BitBtn1.Caption:='CLOSE\LIST';
      end
      else
   begin
    form7.Width:=668;
    form7.Height:=324;
    panel2.Visible:=false;
    DBGrid1.Visible:=false;
    BitBtn1.Caption:='LIST';

    BitBtn2.Enabled:=False;
    DateTimePicker1.Enabled:=false;
    DateTimePicker2.Enabled:=false;
   end;
    end;
```

```
end;

procedure TForm7.BitBtn2Click(Sender: TObject);
Var
  tar1,tar2 : string;
begin
   tar1 := DateToStr(datetimepicker1.date);
   tar2 := datetostr(datetimepicker2.date);
   Query1.Close;
   Query1.SQL.Clear;
   Query1.SQL.Add(format('select * from dosya1 as a where a."date" >= "%s" and
 a."date" <= "%s" ',[tar1,tar2]));
   Query1.Open;
 end;

 procedure TForm7.BitBtn3Click(Sender: TObject);
 begin
 Query1.Close;
 Query1.SQL.Clear;
 Query1.SQL.Add('select * from dosya1');
 Query1.Open;
 end;

  procedure TForm7.Edit4Change(Sender: TObject);
  begin
  Query1.Close;
  Query1.SQL.Clear;
  Query1.SQL.Add('SELECT * FROM DOSYA1');

  Query1.SQL.Add('WHERE REF_NO Like' + QUOTEDSTR(Edit4.Text+'%'));
  Query1.Open;
  end;
```

```
procedure TForm7.Edit3DblClick(Sender: TObject);
begin
edit3.clear;
end;

procedure TForm7.Edit4DblClick(Sender: TObject);
begin
edit4.Clear;
end;

procedure TForm7.Edit2DblClick(Sender: TObject);
begin
edit2.Clear;
end;

procedure TForm7.Edit1DblClick(Sender: TObject);
begin
edit1.Clear;
end;

procedure TForm7.Edit3KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then edit4.SetFocus;
end;

procedure TForm7.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then edit1.SetFocus;
end;

procedure TForm7.Edit4KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then edit2.SetFocus;
```

```
end;

procedure TForm7.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then BitBtn1.SetFocus;
end;

procedure TForm7.janRoundButton2Click(Sender: TObject);
begin
   form14.QuickRep1.Preview;
end;

procedure TForm7.FormCreate(Sender: TObject);
begin
DateTimePicker1.date:=date;
DateTimePicker2.date:=date;
end;

procedure TForm7.DateTimePicker1KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then DateTimePicker2.setfocus;
end;

procedure TForm7.DateTimePicker2KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then edit2.setfocus;
end;
```

```
unit MAIL;

interface

uses
  inifiles, Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, IdComponent, IdTCPConnection, IdTCPClient, IdMessageClient,
  IdSMTP, ComCtrls, StdCtrls, Buttons, ExtCtrls, IdBaseComponent, IdMessage;

type
  TForm9 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    ledHost: TLabeledEdit;
    ledFrom: TLabeledEdit;
    ledTo: TLabeledEdit;
    ledCC: TLabeledEdit;
    btnSendMail: TBitBtn;
    StatusMemo: TMemo;
    ledSubject: TLabeledEdit;
    SMTP: TIdSMTP;
    MailMessage: TIdMessage;
    AttachmentDialog: TOpenDialog;
    Body: TMemo;
    ledAttachment: TLabeledEdit;
    btnAttachment: TBitBtn;
    procedure btnSendMailClick(Sender: TObject);
    procedure btnAttachmentClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure SMTPStatus(ASender: TObject; const AStatus: TIdStatus;
      const AStatusText: String);
  procedure BodyChange(Sender: TObject);

 private
  procedure GetSettings;
   procedure SaveSettings;

 public
   { Public declarations }
 end;

var
  Form9: TForm9;
implementation

{$R *.dfm}
procedure TForm9.btnSendMailClick(Sender: TObject);
begin
  StatusMemo.Clear;
  SMTP.Host := ledHost.Text;
  SMTP.Port := 25;
  MailMessage.From.Address := ledFrom.Text;
  MailMessage.Recipients.EMailAddresses := ledTo.Text + ',' + ledCC.Text;
  MailMessage.Subject := ledSubject.Text;
  MailMessage.Body.Text := Body.Text;
  if FileExists(ledAttachment.Text) then
    TIdAttachment.Create(MailMessage.MessageParts, ledAttachment.Text);
  try
    try
      SMTP.Connect(1000);
      SMTP.Send(MailMessage);
    except on E:Exception do
      StatusMemo.Lines.Insert(0, 'ERROR: ' + E.Message);
    end;
 finally
  if SMTP.Connected then SMTP.Disconnect;
 end;
end;

procedure TForm9.btnAttachmentClick(Sender: TObject);
begin
  if AttachmentDialog.Execute then
   ledAttachment.Text := AttachmentDialog.FileName;
end;
procedure TForm9.FormActivate(Sender: TObject);
begin
 if paramstr(1) <> '' then
  begin
     Update;
     btnSendMail.Click;
     close;
  end;
 end;

 procedure TForm9.FormClose(Sender: TObject; var Action: TCloseAction);
 begin
  SaveSettings;
 end;

 procedure TForm9.FormCreate(Sender: TObject);
 begin
  GetSettings;
 end;

 procedure TForm9.GetSettings;
 var
   ini : TIniFile;
```

```
begin
 Ini := TIniFile.Create(ChangeFileExt(Application.ExeName,'.ini'));
 try
   ledHost.Text := ini.ReadString('SMTP','Host','');
   ledFrom.Text := ini.ReadString('MAIL','From','');
   ledTo.Text := ini.ReadString('MAIL','To','');
   ledCC.Text := ini.ReadString('MAIL','CC','');
   ledAttachment.Text := ini.ReadString('MAIL','Attach','');
   ledSubject.Text := ini.ReadString('MAIL','Subject','')
  finally
   ini.Free;
  end;
 showmessage(paramstr(1)+#13+paramstr(2)+#13+paramstr(3)+#13+paramstr(4)+#13+
 paramstr(5));
   if paramstr(1) <> '' then ledTo.Text := Paramstr(1);
   if paramstr(2) <> '' then ledAttachment.Text := ParamStr(2);
   if paramstr(3) <> '' then ledSubject.Text := Paramstr(3);
   if paramstr(4) <> '' then ledFrom.Text := Paramstr(4);
   if paramstr(5) <> '' then ledCC.Text := Paramstr(5);

  end; (* GetSettings *)

  procedure TForm9.SaveSettings;
  var
    ini : TIniFile;
   begin
    Ini := TIniFile.Create(ChangeFileExt(Application.ExeName,'.ini'));
    try
      ini.WriteString('SMTP','Host',ledHost.Text);


      ini.WriteString('MAIL','From',ledFrom.Text);
      ini.WriteString('MAIL','To',ledTo.Text);
      ini.WriteString('MAIL','CC',ledCC.Text);
  ini.WriteString('MAIL','Attach',ledAttachment.Text);
  ini.WriteString('MAIL','Subject',ledSubject.Text)
 finally
  ini.Free;
 end;
end;

procedure TForm9.SMTPStatus(ASender: TObject; const AStatus: TIdStatus;
  const AStatusText: String);
begin
 StatusMemo.Lines.Insert(0,'Status: ' + AStatusText);
end;
```

CONCULUSION

Archive system is main base registration function of office mamagement for extension. For correct registration of incoming and outgoing files or documents you need always true programming system. Our program is using computer for registration sending and reciving documents from selected office. Documents are enter to the Archive system with their atachments. Any one can easly reach all of enclosed file or documents.

REFERENCES

1-34 Konuda Borland Delphi7 Zeydin Pala

2-www.aboutdelphi.com

3-www.hazirkod.com

4-www.programlama.com

5-www.developershed.com