

**NEAR EAST UNIVERSITY**



**Faculty of Engineering**

**Department of Computer Engineering**

**AIRCRAFT CREW MEMBER PLANNING**

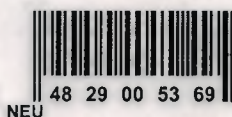
**Graduation Project**

**COM- 400**

**Student: Ishaq Ahmad**

**Supervisor: Assoc. Prof. Dr. Rahib Abiyev**

**Nicosia – 2003**



## ACKNOWLEDGEMENT



*"First of all, I want to pay my regards to my supervisor "Assoc.Prof. Dr. Rahib Abieyv" and all persons who have contributed in the preparation of my project to complete it successfully. I am also thankful to who helped me a lot in my crises and gave me full support toward the completion of my project.*

*I would like to thank my parents who gave their lasting encouragement in my studies, so that I could be successful in my life time. I specially thank to my mother whose prayers have helped me to keep safe from every dark region of life. Special thank to my father who help me in joining this prestigious university and helped me to make my future brighter Special thanks to my friend Alaa whose gave me precious time to help me to complete my study..*

*I am also very much grateful to my all companions and colleagues who gave their precious time to help me to encourage me their ever devotion and all valuable information which I really need to complete my project.*

*Further I am thankful to Near East University academic staff and all those persons who helped me or encouraged me incompletion of my project. Thanks!"*

## ABSTRACT

The purpose of this project is the development of Aircraft Crewmember Planning Database. The analysis of the planning of the crewmembers has been made. The main structures and elements of database system are clarified. The operation principle of each blocks of the system is modeled in Delphi programming language. The developed system allows making decision easily and decreasing time response of the system. Over the past decades people have transferred in maintaining records through paper and pen, and now we are evolving into the technology aria.

This project has taken a lot of effort and time to send out a very clear and simple program in Delphi concerning any airline company. Simplicity was the main object in modifying and keeping records of database concerning airline issues such as crew information and aircraft information .if there is anything that we should agree upon is program save a lot of time and energy.

## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENT</b>	i
<b>ABSTRACT</b>	ii
<b>TABLE OF CONTENTS</b>	iii
<b>INTRODUCTION</b>	1
<b>CHAPTER1: DESCRIPTION OF AIRCRAFT CREW PLANNING</b>	
1.1 Introduction	3
1.2 Records of Flight and Duty Times	3
1.3 Crew Responsibilities	3
1.4 Flight Time Limitation of cockpit crewmembers	4
1.5 Limit terms explanation	4
1.5.1 Rest time	4
1.6 Flight Time Limitation of Cabin crewmembers	5
1.6.1 Rest time	5
1.7 How We Can Calculate the Flight Time, Duty Time and Rest Time	6
1.7.1 Flight Time	6
1.7.2 Duty Time	6
1.7.3 Rest Time:	7
1.8 Crewmembers Duties:	7
1.9 Standby	7
1. 10 Terminology	8
<b>CHAPTER 2: PROGRAM STRUCTURE FOR CREW PLANNING</b>	
2.1 Program Structure	10
2.1.1 Explanation of Block Diagram:	10
2.2 Flow-Chart for Add new flight	12
2.2.1Explanation of the Add new flight flow-charts	17
2.3 Search Flow-Chart	19
2 .3.1 Explanation of the Search	19
<b>CHAPTER 3: DATABASE USING DELPHI</b>	
3.1 What Is Delphi?	20
3.2 Database Basics	20
3.3 Local Databases	22
3.4 Client/Server Databases	22
3.5 The Borland Database Engine	23
3.6 BDE Drivers	23
3.7 BDE Aliases	24
3.8 Creating a BDE Alias	24
3.9 Creating an Alias with the BDE Administrator	25
3.10 Creating an Alias through Code	26



3.11 SQL Links:	27
3.12 DATABASE SERVERS	27
3.13 Single-Tier, Two-Tier, and Multitier Database Architecture	27
3.14 LOCAL INTERBASE	29
3.15 Delphi Database Components	29
3.16 The TDataSet Class	30
3.17 The Table Component	30
3.18 Filters	31
3.19 Using Filters In The Table Component	32
<b>CHAPTER 4: COMPUTER REALIZATION OF CREW PLANNING</b>	
4.1 Database Structure	33
4.2 Master/Detail Tables	36
4.3 Database Form	39
4.3.1 Log-in screen	39
4.3.2 Main Menu Screen	39
4.3.3 Member Information Screen	41
4.3.4 Address Screen	42
4.3.5 License Screen	42
4.3.6 Add new flight Screen	43
4.3.7 Flight Calculation Screen	43
4.3.8 View Crew Screen	44
4.3.9 Members Available screen	44
4.3.10 Aircraft screen	45
4.3.11 Search Screens:	46
4.3.12 Report screen	48
4.3.13 Change Account Screen	48
4.3.14 Add new Account Screen	49
<b>CONCLUSION</b>	<b>50</b>
<b>REFERENCES</b>	<b>51</b>
<b>APPENDIX</b>	<b>52</b>

## INTRODUCTION

This aim of the project is the Aircraft Crewmember Planning using Object- oriented programming technologies.

The intended audience for this project includes the follow:

(1) Codes - any codes that are responsible for creating and maintaining the data elements and file description specified in this project.

(2) Screens - those individuals who wish to view the data collected and processed as part of the Aircraft Crewmember Planning from a "summary" or "subtotaled" point of view. The database files are used by the company staff to verify that the underlying unitary data reported as part of the project are valid and consistent from term-to-term and year-to-year.

In this project, I am trying to send out a message in the shape of a program created in the Delphi language. This message that I am trying to send to you (computer user) is that it is much easier than you think it is to design a program in Delphi for airline companies. Regarding this program which basically, divided into two main sections. A section for the crewmember which consist of cabin chief, senior cabin attendant and cabin attendant and if concentrates on the services and assurance of comfort to their passengers. The other section is the cockpit crew which consists of the captain, first officer and flight engineer. Each member has been assigned a special form in this program; these forms are updated consistently depending on his working hours. In order to clarify this let take an example, if we were to consider a captain X in Y airlines and he was assigned to flight from Ercan to Istanbul , his personal form where his name, age and number of hours flown by him would automatically be updated upon his arrival in Istanbul. This will be very vital information in his resume where he might be needing of any time of his life. Another important thing about this program is it checks for the availability of either the cabin crew or the cockpit crew and by doing so, you wont finds difficulty assigning your crew according to your schedules.

This program capabilities do not end of the cabin and cockpit crew, it extends up to the flight schedules which are linked directly with the aircraft menu.

The flight schedules are typed into the program manually and contain information about the flight number which automatically gives you where the aircraft is heading to in other words its destination and its type, name, capacity and what type of crew is assigned to it. One other information is given about the aircraft in the program is its maintenance and the duration regarding this issue. (Described in chapter 3)

However, the program can be updated in the sense of having its own schedule linked to Delphi program but that would be a question of time., The aircraft section in this program varies in information all linked to its menu. Just like any information about any aircraft in the world, this part of the program is designed for the identify of the aircraft information such as name of aircraft , its type , number of passenger and number of passenger and number of flight crew including a link to cabin and cockpit menu .the aircraft maintenance is also among the information given the type of the aircraft also determine which captain flies which aircraft and which aircraft is assigned to long hour flight and issues like that. another important factor about the aircraft is that depending on number of passenger and flight crew is the aircraft carefully selected .as an example and reference of this program I have used some valuable information from Cyprus Turkish airlines where they have three types of aircrafts which can be flown by any of their cockpit crew. In doing so the flights are almost the same in avionic control



## **CHAPTER 1**

### **DESCRIPTION OF AIRCRAFT CREW PLANNING**

#### **1.1 Introduction**

KTHY has established a Flight and Duty Time limitation and a rest scheme for all crewmembers in accordance to some Rules. The Flight planning for a flight and cabin crew should be planned in accordance with limitations. In the planning process, possible delay chances at the home base and out of the base will be taken into considerations. All flight crewmembers should have the current licenses (no copies) in regard to their types of airplanes and must be available when requested.

For Flight planning the following factors should be considered:

- Command and Control of Flight.
- An accident prevention procedure.
- The usage of communication and navigation system.
- Command and Decision.

#### **1.2 Records of Flight and Duty Times:**

Sufficiently detailed records of crewmember as:

- Block Times.
- Flight Duty Periods.
- Duty Periods.
- Rest periods and local days free of all duties are maintained by the flight operations management to ensure compliance within there requirements. Each crewmember should also keep the same records individually.

#### **1.3 Crew Responsibilities:**

A crewmember shall not operate on an aero-plane if feels unfit to extend that the flight may be endangered.



## 1.4 Flight Time Limitation of cockpit crewmembers:

Flight Time, Duty Time, and Rest Periods must not exceed the following limits:

Table 1.1 Cockpit Limitations

CREW		MAX DUTY TIME	MAX FLIGHT TIME (HOURS)				REST TIME HOURS	
			DAY	WEEK	MONTH	YEAR	DAY	WEEK
A	A310	14	12	30	110	1000	8	48
B	B727	14	12	30	110	1000	8	48
C	B737	14	12	30	110	1000	8	48

## 1.5 Limit terms explanation:

### 1.5.1 Rest time

The minimum rest time be the highest of A, B, C

A) 8 hours

B) 2 x Flight Time

C) 1 x Duty Time

Note:

- Flight time limitations are for 3 landings only. If landing number exceeds, time limits decrease; 1/2 hours for each extra landing.
- None of the Cockpit crewmember must exceed 300 hours flight time in consecutive 90 days.
- Each Flight Duty should not exceed 5 landings.
- In consecutive 7 days at least one continue rest period of 24 hour should be provided to crewmembers.

- A crewmember should have at least 6 days rest period in a month and, at least 4 of them should be consecutive 2 by 2.
- Deadhead Flight, before and after Flight Duty, should be considered Flight Duty hours. Deadhead flight periods to home base as also considered as Flight duty hours for the calculation of rest periods;
- Observer Flight should be considered as Duty periods.

**Table 1.2 Cabin Limitations**

CREW		MAX DUTY TIME	MAX FLIGHT TIME (HOURS)				REST TIME	
			DAY	WEEK	MONTH	YEAR	DAY	WEEK
A	A310	16	12	30	110	1000	8	48
B	B727	16	12	30	110	1000	8	48
C	B737	16	12	30	110	1000	8	48

## 1.6 Flight Time Limitation of Cabin crewmembers:

### 1.6.1 Rest time

The minimum rest time be the highest of A, B, C

- A) 8 hours
- B) 2 x Flight Hours
- C) 1 x Duty Time

- The same limitations are applied to Cabin-crewmembers. However, if required, Flight Duty and Duty Time limits can be increased by % 10, Rest periods remains the same.
- None of the Cabin-crewmembers must exceed 300 hours Flight time in consecutive 90days.

## **1.7 How We Can Calculate the Flight Time, Duty Time and Rest Time:**

There is a Flight leaving from ERCAN at 05:00 to ISTANBUL at 06:30 and then leaves ISTANBUL at 07:30 to ERCAN at 09:00.

ECN 05:00 IST 06:30

IST 07:30 ECN 09:00

### **1.7.1 Flight Time:**

The flight time from ECN to IST:  $05:00 - 06:30 = 01:30$

The flight time from IST to ECN:  $07:30 - 09:00 = 01:30$

The total flight time of this flight:  $01:30 + 01:30 = 3 \text{ hours}$

### **1.7.2 Duty Time:**

In order to calculate the Duty Time certain steps are made which are the following:

A) Subtracting 1 hour from the first Departure Time.

B) Adding 30 minutes to the last Landing Time.

C) Duty time = Subtract the result of A from the result of B.

$05:00 - 01:00 = 04:00$

$09:00 + 00:30 = 09:30$

Duty Time:  $09:30 - 04:00 = 05:30 \text{ hours}$ .

### **1.7.3 Rest Time:**

The minimum Rest Time is the highest of A, B and C.

A) 8 hours

B)  $2 \times \text{Flight Time}$

C)  $1 \times \text{Duty Time}$

Rest Time will be = 8 hours.

## 1.8 Crewmembers Duties:

**Table 1.3** Crewmember Duties

AIRCRAFT	COCKPIT	CABIN
B-727	1 CAPTAIN 1 FIRST OFFICE 1 FLIGHT ENGINEER	1 CABIN CHIEF 1 SENIOR CABIN 2 CABIN ATTENDANTS
B-737	1 CAPTAIN 1 FIRST OFFICE	1 CABIN CHIEF 1 SENIOR CABIN 2 CABIN ATTENDANTS
A-310	1 CAPTAIN 1 FIRST OFFICE	1 CABIN CHIEF 2 SENIOR CABIN 3 CABIN ATTENDANTS

## 1.9 Standby:

**Table 1.4** Standbys

STANBY TYPE	STANBY HOURS
Standby 1	Between 00:00 to 12 :00
Standby 2	Between 12:00 to 00:00

All standby crews will stay at home or a suitable place where they can be contacted directly by the crew planning department or operation control center. When called by the company, a crewmember on standby will be at KTHY operation office within reasonable time.

## 1.10 Terminology:

*Actual Flight Operation:* Actual flight operation starts at the reporting time and ends when the crew goes off Duty.



*Adequate Facilities:* A quiet and comfortable place not opens to the public.

*Augmented flight Crew:* A flight crew which comprises more then the minimum number required for the operation of the aero plane and in which each flight crew member can leave his post and be replaced by another appropriately qualified flight crew member.

*Flight Duty Period (FDB) (Duty Time):* A period starting when the crew member is required to report for a duty period which includes a flight, and which finishes at the block time at the end of the final flight on which the crew member Is an operating crewmember One hour before scheduled of block time until 1/2 hour after on block time.

*Home Base:* The KTHYis normally nominated to the crew member ERCAN from Where the Crewrnmber starts and ends a duty period or a series of duty periods and not responsible for the accommodation of the crewmember consumed.

*Late Finishes / Warty starts:* Any duty that is carried out within any part of the period 0100 to 0659 hours local time at the reporting place.

*Local Day:* A period from 0000 to 2400 local time.

*Notification Time:* The period of time that KTHY allows between the time a crew member on standby receives a call requiring him/her to report for duty and the time he/she is required to report for that duty.

*Operation Crewmember:* A crewmember who carries out his /her duties in an aero plane during the flight or during any part of it.

*Reporting Time:* The time at which a crewmember is required by KTHY to report for any duty.

*Rest Period:* An uninterrupted and defined period of time during which a crewmember is free of all duties and be standby. (Start 1/2 hour after on Block time).

*Split Duty:* Flight duty periods, which consist of two duties, separated by a break.

*Standby:* A period of time in which a crewmember has not been assigned to any duty, but is required to be contactable to receive an assignment for duty.

**Suitable Accommodation:** A suitably finished bedroom, with single occupancy, if required by the crew member, which is subject to minimum noise, is well ventilated and should have the facility to control the levels of light and temperature.



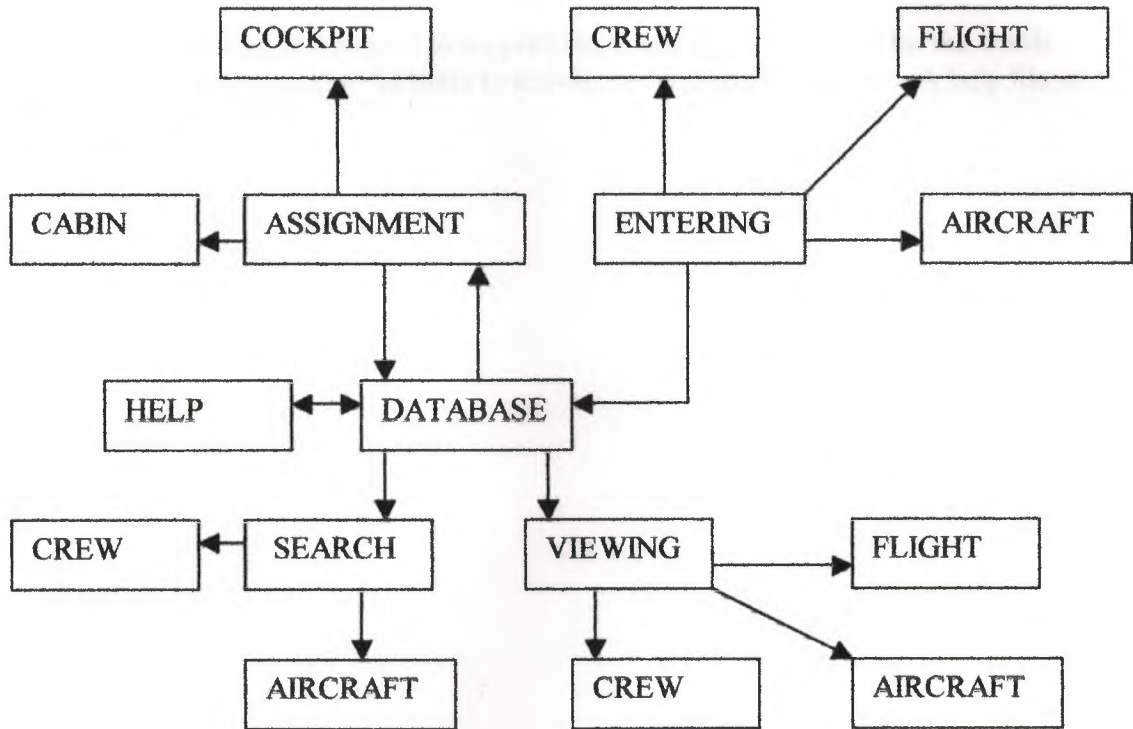
## 2.1.2 Representation of Block Diagrams

- \* Program/Block Diagrams contain information on the design of the system (as a whole) and the information is organized in a certain way, which is the basis for the system design.
- \* Block Diagrams contain information on the design of the system as a whole.
- 1. General information
- 2. Information on the system
- 3. Information on the system

## CHAPTER 2

### PROGRAM STRUCTURE FOR CREW PLANNING

#### 2.1 Program Structure:



##### 2.1.1 Explanation of Block Diagram:

- **Assignment Block:** we enter certain information in order to assign the Crewmember (he / she) whether be Cabin or Cockpit in a certain Flight ,after so ,the information will be stored in the database file.
- **Entering Block:** for this block entered information would be chosen due to three data types which are as the following:
  1. Crewmember information
  2. Flight Schedule Information
  3. Aircraft Information

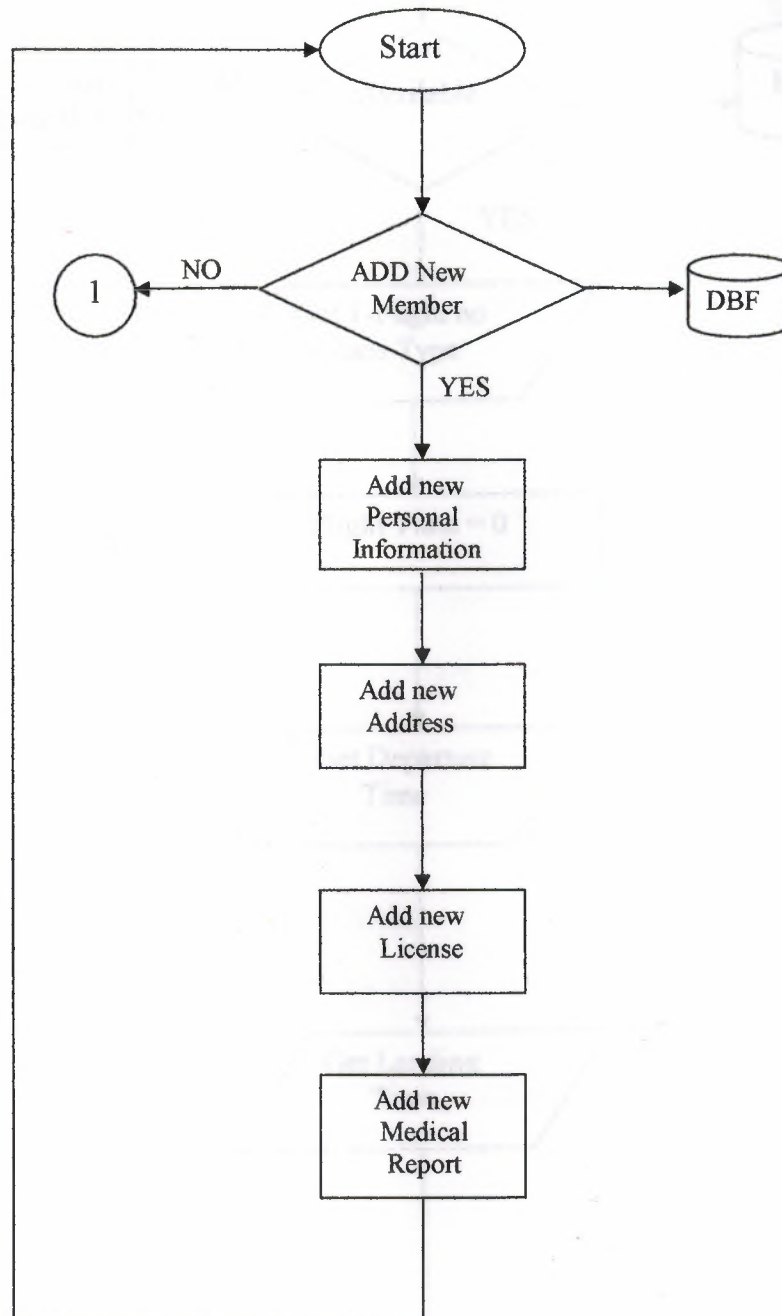
After choosing the data type; entered information, will be stored in the database file.

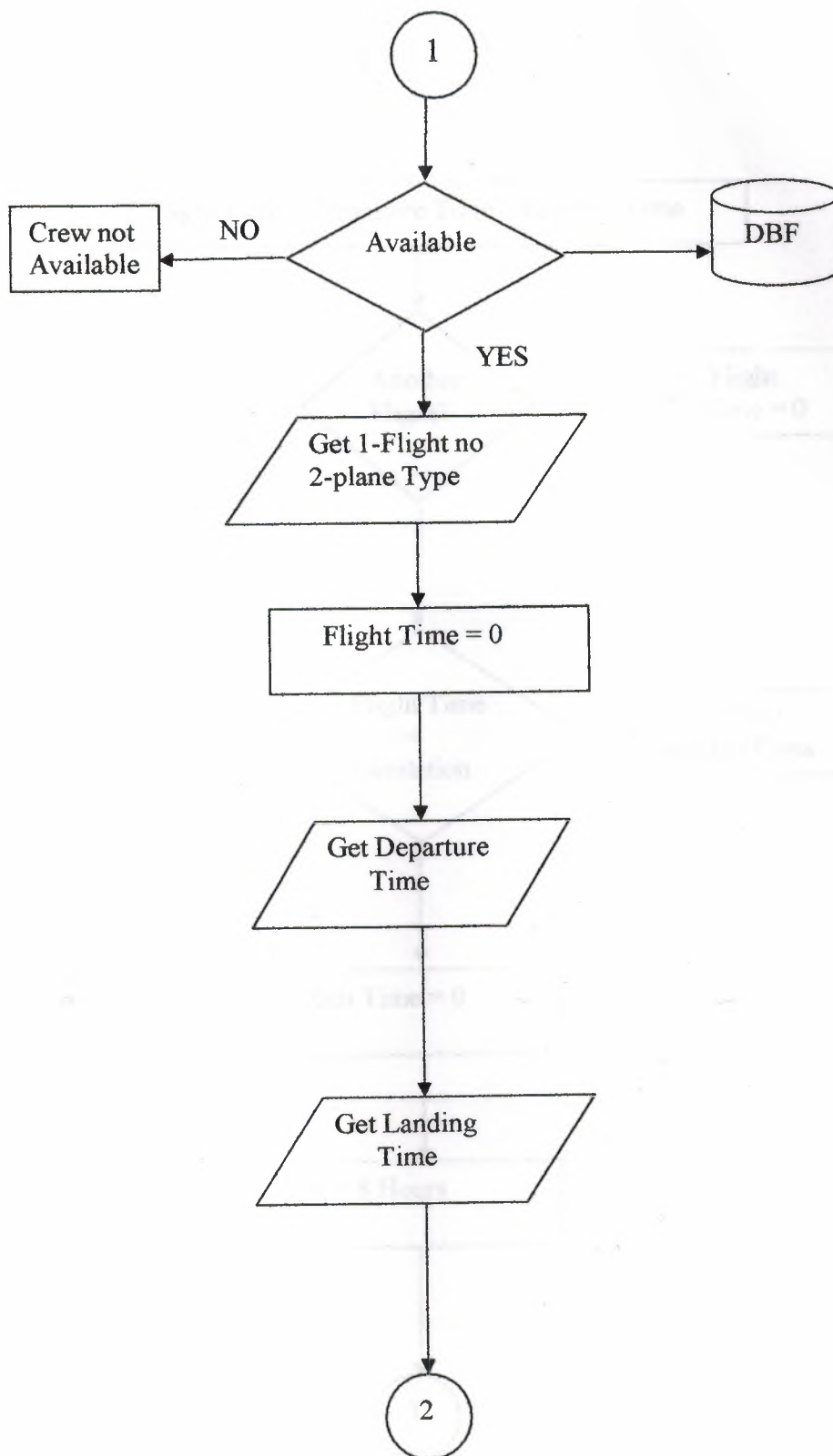
- **Viewing Block:** This block shows all the information that has been entered about the crewmembers, cabin, cockpit, aircraft and flight schedule.
- **Search Block :** we use this block to search for Crewmember ( he/she) is cabin or cockpit and we use it to search for aircrafts as well .For this task the program gets data from the user compares it with the database, if the program finds it, then the rest of the data information will appear.
- **Help Block:** this block is designed to support the usage of the operators for the whole program. Unless you are going to be there to answer their question in person. A help file is provided for that application.

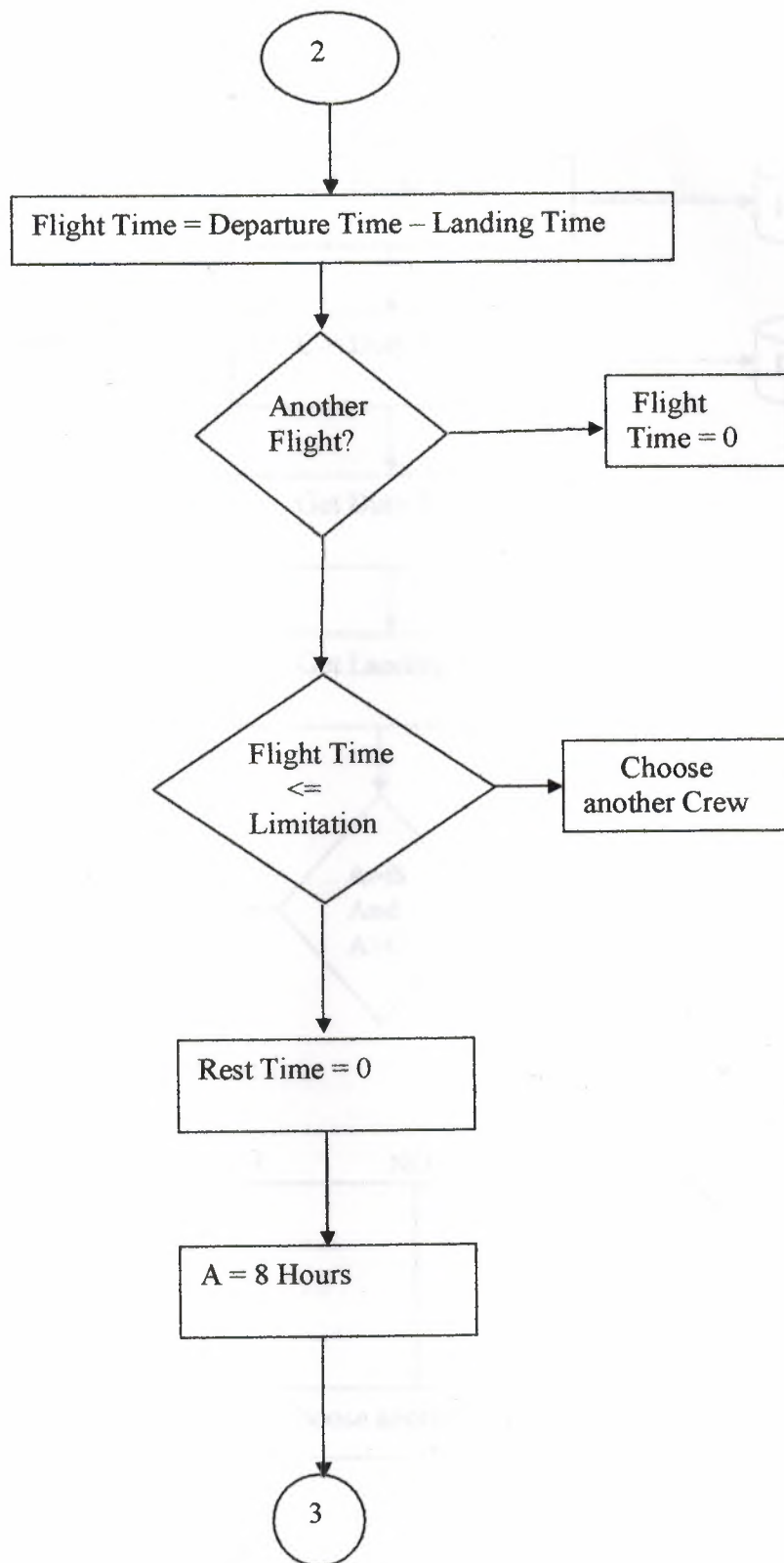


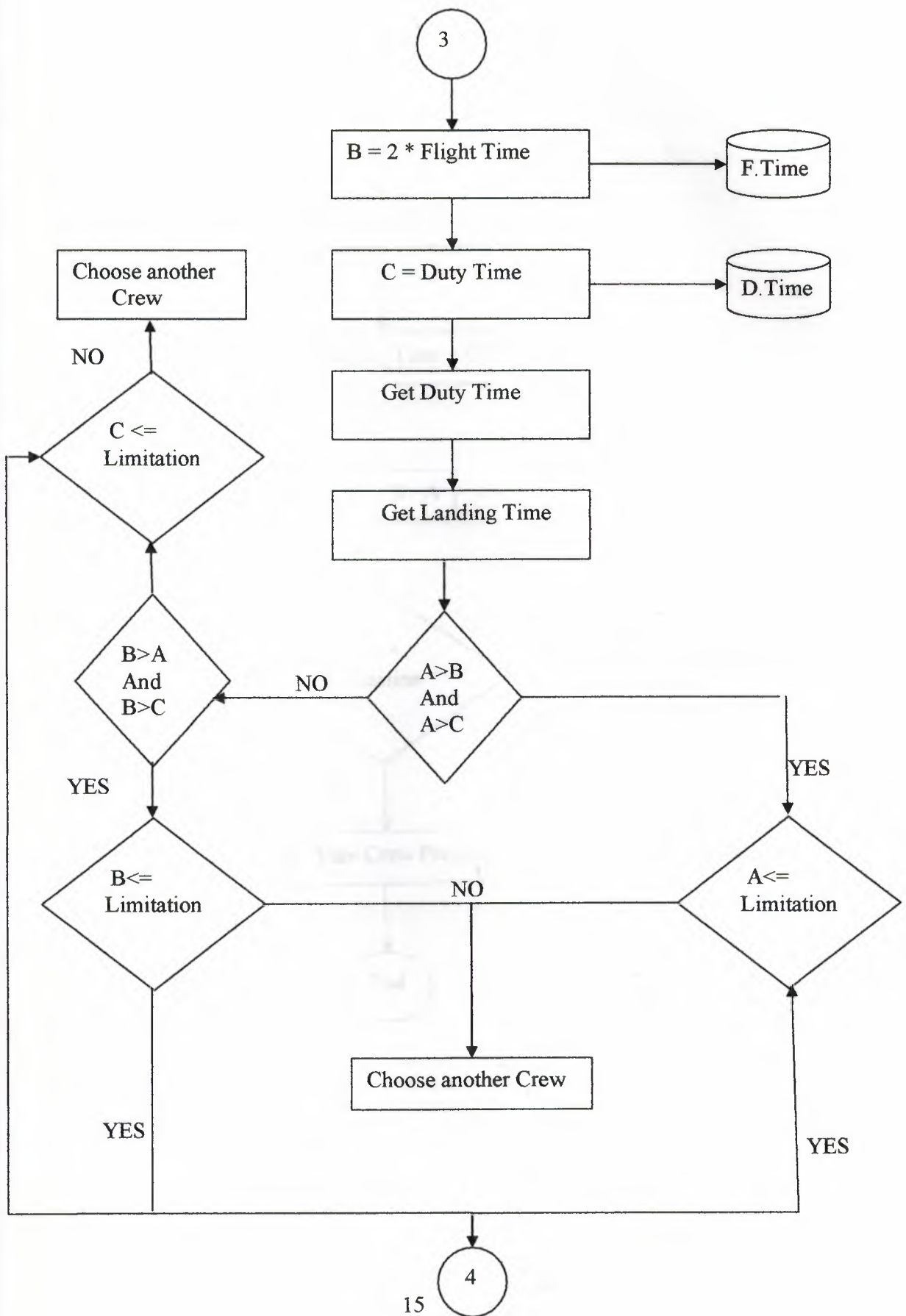


## 2.2 Flow-Chart for Add new flight:

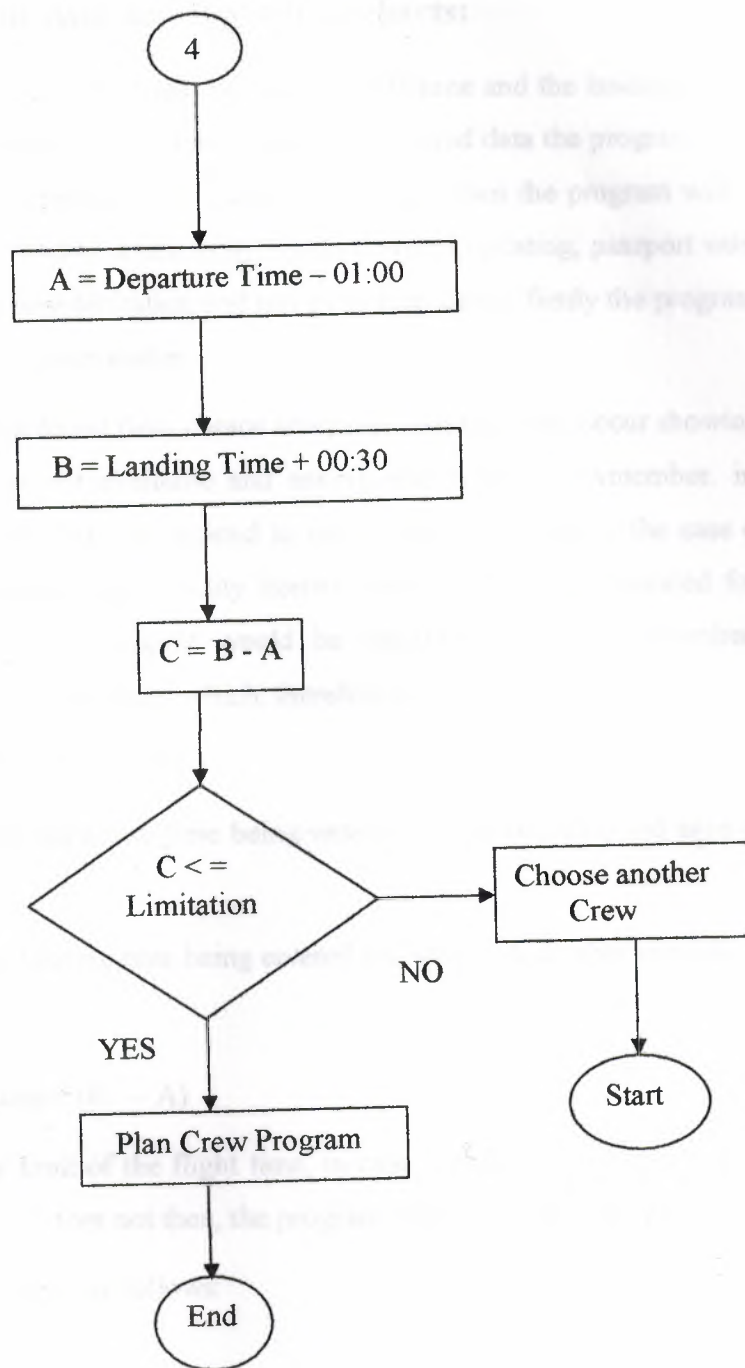












### 2.2.1 Explanation of the Add new flight flow-charts:

At the first the user should enter the flight number, takeoff time and the landing time ,then enter the plane type and the crew type as well due to the entered data the program will start assigning the crewmember whether it is a cabin or a cockpit then the program will check the crewmember's license validity, availability, medical report updating, passport validity , flight time limitation, duty time limitation and rest period limitation firstly the program will check the availability of the crewmember

From the database, since not found (less choice able) alert message will occur showing that the desired crewmember is not available and asking into a new crewmember. in this message alerting system will show in respond to incomplete conditions in the case of the more avoidably found crewmember .validity license check will be demonstrated for that crewmember therefore a valid passport would be required for the crewmember The calculation of the flight time is next step which, therefore as

1-assuming the flight time = 0 (constant)

2-The program will get the departure time being entered by the operator and save it in a variable (A).

3-The program will get the landing time being entered and save it in another variable (B)

4-the flight time = flight time + (B — A)

5-After so, will check the limit of the flight time, in case it passed over the limitation the alert message will occur, if it does not then, the program will check the Duty time.

The calculation of the duty time as follows:

1-assuming A = 8 hours

2-get the flight time of the crewmember from the database and multiplied by 2, then save In a variable (B)

3-Get the duty time of the crewmember from the database and save it in another variable

4-The program will take the highest value of those variables and save it as the rest time, and then compare the rest time with limitation

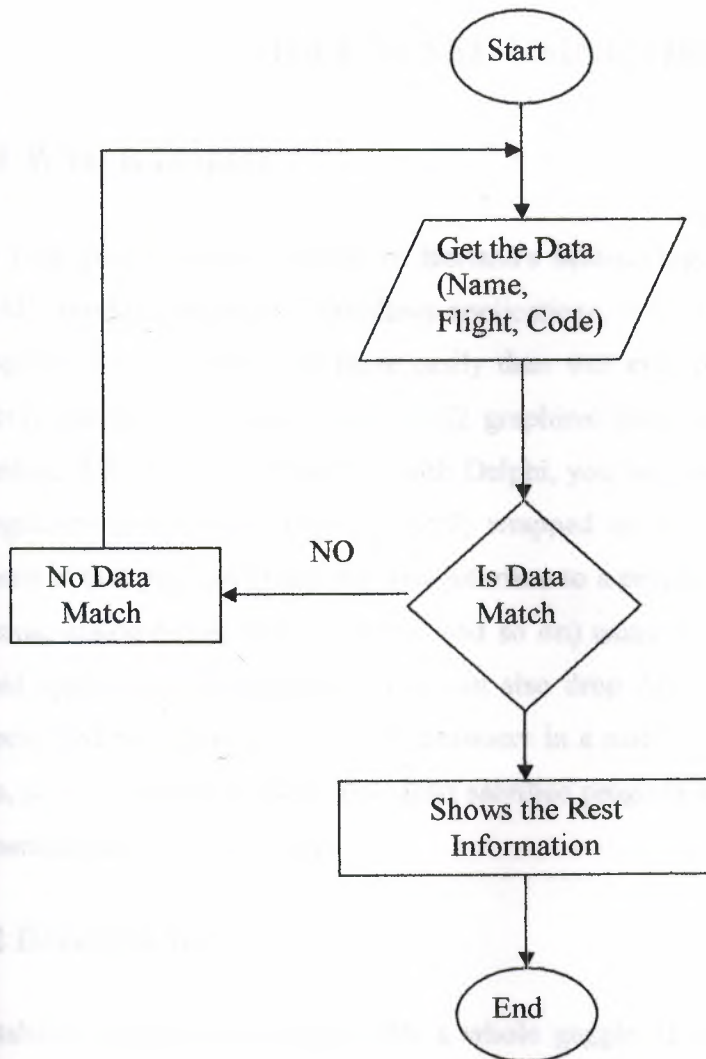
after that the program will check the last condition which is the duty time, the program will subtract one hour from departure time and save it in a variable (A), then the program will add 30 minutes into the landing time and save the result in a variable (B), finally subtract B from A, if the result of the subtraction passed over the limit the alert message will occur other wise the crew will be assigned.



### 2.3.1 Validation and the Search:

The validation and the search in the program to get the data will be done by the user, where the user will enter the flight number, the date, and the code, if the data is not correct, the program will show an alert message, and if the data is correct, the program will show the rest time and the crew information.

### 2.3 Search Flow-Chart:



#### 2 .3.1 Explanation of the Search:

The required data which is entered to the program to search for will be checked to know whether the data is matched or not according to the database. If the data is not matched then alert message will occurs otherwise the program continue the process that shows the rest information.



## CHAPTER 3

### DATABASE USING DELPHI

#### 3.1 What is Delphi?

By now you know that Delphi is Borland's best-selling rapid application development (RAD) product for writing Windows applications. With Delphi, you can write Windows programs more quickly and more easily than was ever possible before. You can create Win32 console applications or Win32 graphical user interface (GUI) programs. When creating Win32 GUI applications with Delphi, you have all the power of a true compiled programming language (Object Pascal) wrapped up in a RAD environment. What this means is that you can create the user interface to a program (the *user interface* means the menus, dialog boxes, main window, and so on) using drag-and-drop techniques for true rapid application development. You can also drop ActiveX controls on forms to create specialized programs such as Web browsers in a matter of minutes. Delphi gives you all this, and at virtually no cost: You don't sacrifice program execution speed because Delphi generates fast compiled code.

#### 3.2 Database Basics:

Database programming comes with a whole gaggle of buzzwords: *BDE*, *client*, *server*, *ODBC*, *alias*, *SQL*, *query*, *stored procedure*, and so on. The good news is that it isn't all that bad after you learn some basics. First, let's take a moment to talk about databases. When you hear the word *database*, you probably imagine data stored in table format. The table probably contains fields such as *FirstName*, *LastName*, and *PhoneNumber*. These fields are filled with data to create individual records in a database file.

If that's what you envision when you think of a database, you're not too far off, but you aren't exactly correct, either. The term *database* is used to describe an all-encompassing data creation and maintenance system. It is true that a database can be as simple as one table. On the other hand, a real-world database can include dozens or even hundreds of

## CHAPTER 3

### DATABASE USING DELPHI

#### 3.1 What is Delphi?

By now you know that Delphi is Borland's best-selling rapid application development (RAD) product for writing Windows applications. With Delphi, you can write Windows programs more quickly and more easily than was ever possible before. You can create Win32 console applications or Win32 graphical user interface (GUI) programs. When creating Win32 GUI applications with Delphi, you have all the power of a true compiled programming language (Object Pascal) wrapped up in a RAD environment. What this means is that you can create the user interface to a program (the *user interface* means the menus, dialog boxes, main window, and so on) using drag-and-drop techniques for true rapid application development. You can also drop ActiveX controls on forms to create specialized programs such as Web browsers in a matter of minutes. Delphi gives you all this, and at virtually no cost: You don't sacrifice program execution speed because Delphi generates fast compiled code.

#### 3.2 Database Basics:

Database programming comes with a whole gaggle of buzzwords: *BDE*, *client*, *server*, *ODBC*, *alias*, *SQL*, *query*, *stored procedure*, and so on. The good news is that it isn't all that bad after you learn some basics. First, let's take a moment to talk about databases. When you hear the word *database*, you probably imagine data stored in table format. The table probably contains fields such as *FirstName*, *LastName*, and *PhoneNumber*. These fields are filled with data to create individual records in a database file.

If that's what you envision when you think of a database, you're not too far off, but you aren't exactly correct, either. The term *database* is used to describe an all-encompassing data creation and maintenance system. It is true that a database can be as simple as one table. On the other hand, a real-world database can include dozens or even hundreds of

tables with thousands or millions of records. These tables can contain one or more indexes. A complete client/server SQL database solution can also contain numerous queries and stored procedures. (Don't worry; I'll explain some of these terms later in the chapter.) So as you can see, a database is more than just a table with data.

Speaking of tables, let's quickly cover some table basics. A table consists of at least two parts: fields and records. *Fields* are the individual categories of data in a table. For example, a table containing an address book would have a field called `FirstName`, a field called `LastName`, one called `Address`, `PhoneNumber`, and so on. Fields are also referred to as *columns*. A record, then, is one person's complete address: first name, last name, address, and so on. Records are also called *rows*.

A database is just a collection of data, of course, but database tables are often displayed in spreadsheet format. The column headers across the top indicate the field names. Each row in the table contains a complete record. Figure 16.1 shows just such a database table

Displayed in grid (or table) format.

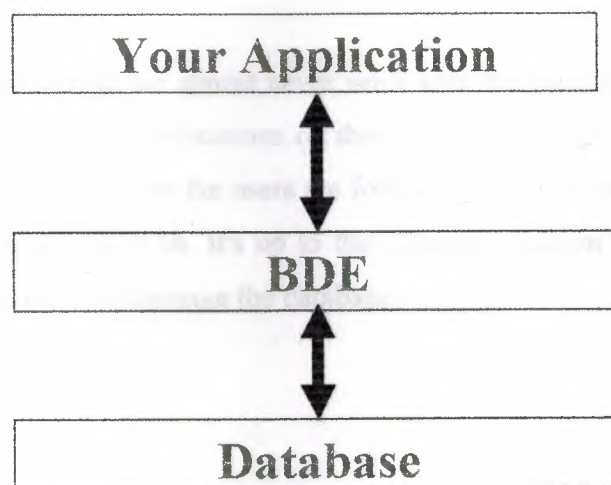


Figure 3.1 a typical database table



### 3.3 Local Databases:

The simplest type of database is the local database. A *local database* is a database that resides on a single machine. Imagine that you have a program that needs to store a list of names and addresses. You could create a local database to store the data. This database would probably consist of a single table. The table is accessed only by your program; no one else has access to it. Any edits made to the database are written directly to the database. Paradox, dBase, and Access databases are usually local databases.

### 3.4 Client/Server Databases:

Another way a database can be implemented is as a *client/server database*. The database itself is stored and maintained on a file server (the *server* part of the equation). One or more users (the *clients*) have access to the database. The users of this type of database are likely to be spread across a network. Because the users are oblivious to one another, more than one might attempt to access the database at the same time. This isn't a problem with client/server databases because the server knows how to handle all the problems of simultaneous database access.

The users of a client/server database almost never work with the database directly. Instead, they access the database through applications on their local computer. These applications, called *client applications*, ensure that the users are following the rules and not doing things to the database that they shouldn't be. It's up to the client application to prevent the user from doing something that would damage the database.

### 3.5 The Borland Database Engine:

To enable access to local databases and to client/server databases, Delphi provides the Borland Database Engine (BDE). The BDE is a collection of DLLs and utilities that enables access to a variety of databases.

To talk to client/server databases, you must have the Client/Server version of Delphi. This version ships with SQL Links drivers used by the BDE to talk to client/server databases. Figure 16.2 shows the relationship between your application, the BDE, and the database.

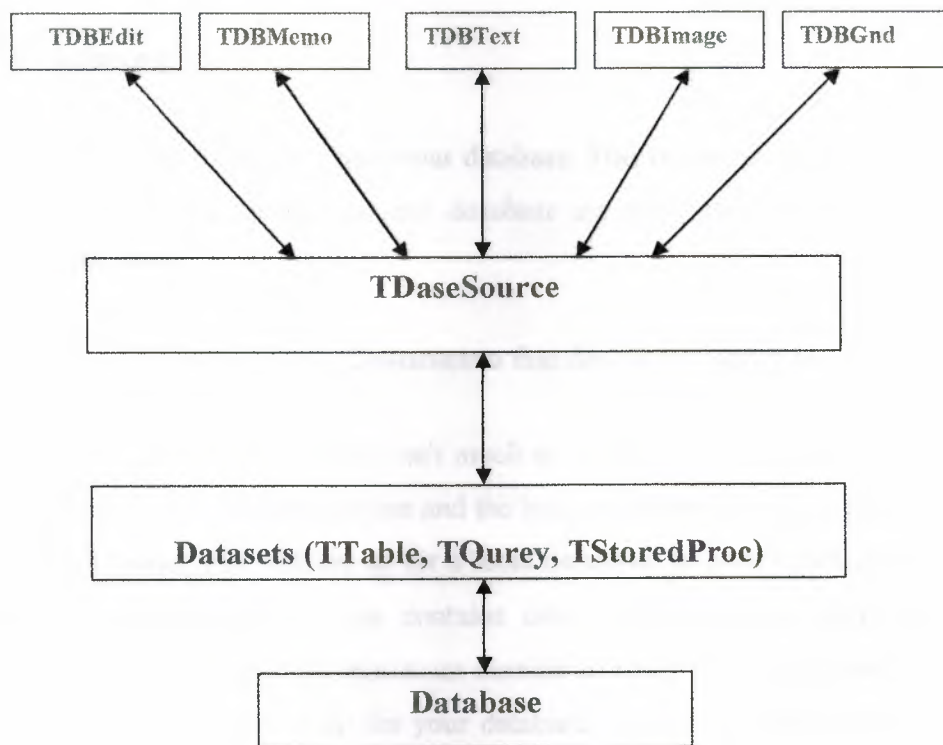


Figure 3.2 your application, the BDE, and the database.

### 3.6 BDE Drivers:

Naturally, database formats and APIs vary widely. For this reason the BDE comes with a set of drivers that enables your application to talk to several different types of databases. These drivers translate high-level database commands (such as open or post) into



commands specific to a particular database type. This permits your application to connect to a database without needing to know the specifics of how that database works.

The drivers that are on your system depend on the version of Delphi you own. All versions of Delphi come with a driver to enable you to connect to Paradox and dBASE databases. This driver, called STANDARD, provides everything you need to work with these local databases.

The Client/Server version of Delphi includes drivers to connect to databases by Sybase, Oracle, Informix, InterBase, and others.

### **3.7 BDE Aliases:**

The BDE uses an alias to access a particular database. This is one of those terms that might confuse you at first. The terms *alias* and *database* are often used interchangeably when talking about the BDE.

**New Term:** A *BDE alias* is a set of parameters that describes a database connection.

When it comes right down to it, there isn't much to an alias. In its simplest form, an alias tells the BDE which type of driver to use and the location of the database files on disk. This is the case with aliases you will set up for a local database. In other cases, such as aliases for client/server databases, the alias contains other information as well, such as the maximum size of BLOB data, the maximum number of rows, the open mode, or the user's username. After you create an alias for your database, you can use that alias to select the database in your Delphi programs. Later today, in the section "Creating a BDE Alias," I'll tell you how to go about creating a BDE alias for your own databases.

### **3.8 Creating a BDE Alias:**

You can go only so far in database programming without eventually creating a BDE alias. The sample databases are fine, but sooner or later you will need to create an alias for your own databases. When you deploy your Delphi database application, you will also need to

create one or more aliases on your users' machines as well. There are many ways to create an alias:

- Through the BDE Administrator utility from the Delphi program group
- Through the Database Desktop program
- Through the SQL Explorer (Client/Server version only)
- Through code at runtime

To create an alias, either you must have your users run the BDE Administrator, or you must create any needed aliases through code. Obviously, creating the alias yourself through code is preferable (never underestimate the ability of your users to botch even the most simple tasks). First I'll show you how to use the BDE Administrator to create an alias. Then I'll show you how to create an alias through code.

### **3.9 Creating an Alias with the BDE Administrator:**

While you are developing your applications, you need to create one or more BDE aliases. This is most easily done using one of the BDE utility programs provided with Delphi. The steps for creating an alias using the BDE Administrator and the SQL Explorer are identical, so for simplicity's sake I'll show you how to create an alias with the BDE Administrator.

Let's assume for a minute that you are going to create a mailing list application. The first step you need to take is to create an alias for your database. You can create an alias in several ways, but the easiest is probably with the BDE Administrator utility. Perform these steps:

1. Start the BDE Administrator (locate the Delphi group from the Windows Start menu and choose the BDE Administrator icon). The BDE Administrator will start and show a list of database aliases currently installed.
2. Choose Object|New from the BDE Administrator menu (make sure the Databases tab is selected). The New Database Alias dialog box comes up and asks which driver to use for the new alias.



3. You will be creating a database using the Standard driver, and because STANDARD is already selected, you can simply click OK. Now the BDE Administrator

4. The BDE Administrator is waiting for you to type a name for your alias, so type MyDatabase and press Enter.

At this point, you need to provide a few items of information in the Definition window. The Type is already set to STANDARD, so there's nothing to be done there. The DEFAULT DRIVER field is set to PARADOX, which is the type you want, so there's nothing to be done there, either (other choices include dBASE, FOXPRO, and ASCII DRV). You can also leave the default value for the ENABLE BCD field. The only information you need to supply is the path on disk where the database files will be stored:

1. Click on the PATH field and either type a path or use the ellipsis button to browse to a path.

2. Close the BDE Administrator and say Yes when asked whether you want to save your edits. That's it. You have created a BDE alias.

Switch back to Delphi and drop a Table component on a form. Check the DatabaseName property in the Object Inspector to see whether your database alias shows up. If you did everything right, you will see it listed there with the other database names. Your database doesn't have any tables yet, but that's okay. You can take care of that later.

### 3.10 Creating an Alias through Code:

To avoid confusion with your users, you will probably want to create any aliases your program needs the first time your program runs. Thankfully, creating an alias at runtime is simple. Here's the code to create a local Paradox alias called WayCool:

```
CreateDirectory('C:', nil);  
Session.AddStandardAlias('WayCool', 'C:', '');
```

That's it? Yes, that's all there is to it. Naturally, you should perform some checks to ensure that the directory and alias were properly created, but that's about all there is to it.

### 3.11 SQL Links:

The Client/Server version of Delphi comes with SQL Links in addition to the BDE. SQL Links is a collection of additional drivers for the BDE. These drivers enable Delphi applications to connect to client/server databases such as those provided by Oracle, InterBase, Informix, Sybase, and Microsoft. Details regarding deployment of SQL Links drivers are also available in `DEPLOY.TXT`.

### 3.12 DATABASE SERVERS:

As long as I am talking about client/server databases, let's take a moment to talk about database servers. Database servers come in several flavors. Some of the most popular include offerings from InterBase (a Borland-owned company), Oracle, Sybase, Informix, and Microsoft. When a company purchases one of these database servers, it also purchases a license that enables a maximum number of users to access the database server. These licensed users are often referred to as *seats*. Let's say a company buys InterBase and purchases licenses for 50 seats. If that company grows to the point that 75 users require access to the database, that company will have to buy an additional 25 seats to be in compliance with the license. Another way that client/server databases are sold is on a *per connection* basis. A company can buy a license for 50 simultaneous connections. That company can have 1,000 users of the database, but only 50 can be connected to the database at any one time. The database server market is big business, no question about it.

### 3.13 Single-Tier, Two-Tier, and Multitier Database Architecture:

Local databases are often called single-tier databases. A *single-tier database* is a database in which any changes--such as editing the data, inserting records, or deleting records--happen immediately. The program has a more direct connection to the database.

In a *two-tier database*, the client application talks to the database server through database drivers. The database server takes the responsibility for managing connections, and the client application is largely responsible for ensuring that the correct information is being



written to the database. A fair amount of burden is put on the client application to make sure the database's integrity is maintained.

In a *multitier client/server architecture*, the client application talks to one or more application servers that, in turn, talk to the database server. These middle-level programs are called *application servers* because they service the needs of the client applications. One application server might act as a data broker, responding to and handling data requests from the client and passing them on to the database. Another application server might only handle security issues.

Client applications run on local machines; the application server is typically on a server, and the database itself might be on another server. The idea behind the multitier architecture is that client applications can be very small because the application servers do most of the work. This enables you to write what are called *thin-client* applications.

Another reason to use a multitier architecture is management of programming resources. The client applications can be written by less experienced programmers because the client applications interact with the application server that controls access to the database itself. The application server can be written by more experienced programmers who know the rules by which the database must operate. Put another way, the application server is written by programmers whose job is to protect the data from possible corruption by errant client applications.

Although there are always exceptions, most local databases make use of the single-tier architecture. Client/server databases use either a two-tier or a multitier architecture.

So how does this affect you? Most applications you write with Delphi for use with a client/server database will be client applications. Although you might be one of the few programmers given the task of writing server-side or middle-tier applications, it's a good bet that you will write primarily client applications. As an application developer, you can't talk directly to these database servers. Let's look next at how a Delphi application talks to a database.



### **3.14 LOCAL INTERBASE:**

The Standard and Professional versions of Delphi come with a single-user copy of Local InterBase. Local InterBase is just what its name implies: a version of InterBase that operates on local databases. The Client/Server version of InterBase, on the other hand, is a full-featured client/server database. The main reason that Delphi ships with Local InterBase is so that you can write an application that operates on local databases and then later change to a client/server database with no programming changes. This gives you an opportunity to hone your client/server programming skills without spending the money for a client/server database.

If you attempt to access a Local InterBase table at either design time or runtime, you will be prompted for a username and password. The Local InterBase administrator is set up with a username of SYSDBA and a password of masterkey. You can use these for login, or you can go to the InterBase Server Manager utility and add yourself as a new user to the InterBase system.

### **3.15 Delphi Database Components:**

Okay, so the preceding section isn't exactly the type of reading that keeps you up all night turning pages. Still, it's important to understand how all the database pieces fit together. With that background, you can now turn your attention to the database components provided by VCL and how those components work together to create a database application. First, I'll give you a quick overview of the VCL database components, and then you'll look at individual classes and components in more detail.

The VCL database components fall into two categories: nonvisual data access components and visual data-aware components. Simply put, the nonvisual data access components provide the mechanism that enables you to get at the data, and the visual data-aware components enable you to view and edit the data. The data access components are derived from the TDataSet class and include TTable, TQuery, and TStoredProc. The visual data-aware components include TDBEdit, TDBListBox, TDBGrid, TDBNavigator, and more. These components work much like the standard edit, list box, and grid components except

that they are tied to a particular table or field in a table. By editing one of the data-aware components, you are actually editing the underlying database as well.

---

**NOTE:** All the VCL database components can be termed data components. I use the term *data access components* for the nonvisual database components on the Data Access tab of the Component palette and the term *data-aware components* for the visual database components from the Data Controls tab.

---

### 3.16 The TDataSet Class:

TDataSet is the ancestor class for TTable, TQuery, and TStoredProc. As such, most properties, methods, and events that these classes use are actually defined by TDataSet. Because so many characteristics of the derived classes come from TDataSet, I'll list the primary properties, methods, and events of TDataSet here, and later I'll list the properties, methods, and events particular to each derived class.

### 3.17 The Table Component:

The Table component, represented by the TTable class, provides the quickest and simplest access to a table. Tables are more than adequate for most single-tier database applications. Usually, you will use the Table component when dealing with local databases and the Query component when dealing with SQL database servers.

The TTable class has many properties and methods in addition to those in its ancestor class, TDataSet. Table 16.4 lists the primary properties of the TTable component and Table 16.5 lists the primary methods. Remember, these are properties and methods specific to TTable and do not include those of TTable's ancestor, TDataSet.

For the most part, the properties and methods are very intuitive. By that I mean that you can usually figure out what a property or method does by just looking at its name. It doesn't take a lot to figure out that the LockTable method locks a table for an application's specific



use and that the UnlockTable method unlocks the table again. Likewise, you don't have to have an IQ of 150 to guess what the CreateTable, DeleteTable, and RenameTable methods do. With that in mind, I'm not going to cover every aspect of every property and method listed here. Instead, let's get on to some of the more interesting aspects of the Table component.

**TABLE 3.1 Primary TTable Properties**

<i>Property</i>	<i>Description</i>
Exclusive	Locks a local table so that only this application can use it.
IndexDefs	Contains information about the table's indexes.
IndexFieldCount	The number of fields that make up the current key.
IndexFieldNames	Used to set the current key by specifying the names of the fields to use for the index.
IndexFields	Used to retrieve information about a specific field in an index.
IndexName	Used to specify a secondary index for a table.
KeyFieldCount	The number of fields to use when searching on partial keys.
MasterFields	The field or fields that should join the master and detail tables.
MasterSource	The table to be used as a master table when this table is used as a detail table.
ReadOnly	Specifies whether this table is read-only.
TableName	The name of the database table.
TableType	The table's type (Paradox, dBASE, or ASCII).

### **3.18 Filters:**

A common need of a database application is to filter a table. Before I discuss filters in detail, I want to point out that filters are primarily used on local databases. Filters are rarely used with client/server databases; instead, a SQL query is used to achieve the same effect that filters have on local databases.

So why filter? Consider that you might have a table with thousands of records, but you are interested in displaying or working on only a small subset of the table. Let's say you have a database that contains names and addresses of computer users all over the world. Your company sells these names and addresses to other companies that want to do bulk mailings.

I call and want to order a mailing list from your company, but I want the list to contain only those computer users who live in Colorado. You could filter your table by postal code and generate a list of names with only Colorado addresses. Or, maybe Borland calls you and wants a list of computer users in Great Britain who are programmers by occupation. In that case, you could filter by occupation and country, thereby giving only the names and addresses the customer is interested in.

### **3.19 Using Filters in the Table Component:**

Filters in the Table component are handled in one of two ways: through the Filter property or the OnFilterRecord event. Before I discuss these, let me talk about the Filtered property. This property determines whether the table is filtered. If Filtered is True, the table will apply the filter currently in force (either the contents of the Filter property or the results of the OnFilterRecord event). If Filtered is False, the contents of the Filter property are ignored and the OnFilterRecord event is never generated.

## CHAPTER 4

### COMPUTER REALIZATION OF CREW PLANNING

#### 4.1 Database Structure:

First thing to know about Delphi in the relational database model, how to use the Paradox database objects to access and update existing databases, and how to use the program to create and maintain databases. We also take a look at the design data entry and how to create input validation routines at the keystrokes, field, and form levels. Taking in mind that Paradox is the best way to make the relations and build up the main structure that the Delphi depends upon. And below there are some database tables which I used in this program:

**Table 4.1** Main Table

Field Name	Type	Size	Key
FlightNo	N		*
Pilot	A	15	
FlightOfficer	A	15	
flightEngineer	A	15	
CabinChief	A	15	
SeniorCabin1	A	15	
SeniorCabin1	A	15	
PartTimeHosees1	A	15	
PartTimeHosees2	A	15	
Flight	A	20	
Date	D		
DepTime	T		
ArrTime	T		
PlaneName	A	10	



**Table 4.2 Flight**

Filed Name	Type	Size	Key
Flight code	A	10	*
Plane	A	6	
ID	A	10	
DepAir	A	5	
DepTime	T		
ArrAir	A	5	
AirTime	T		

**Table 4.3 Airport**

Filed Name	Type	Size	Key
Capacity of air	A	10	
planeName	A	10	
Country	A	10	
Air_code	A	10	
Available	A	10	

**Table 4.4 personal Details**

Field Name	Type	Size	Key
Code	A	5	*
Name	A	10	
Surname	A	10	
DateOfBirth	D		
PlaceOfBirth	A	15	
NameOfMother	A	10	
Gender	A	5	
PassportNo	A	20	
Married	L		
Nationality	A	15	
Duty	A	15	
Picture	G		

**Table 4.5 Duty**

Filed Name	Type	Size	Key
Duty code	A	3	*
Duty	A	20	

**Table 4.6 License**

Filed Name	Type	Size	Key
Code	A	5	*
License No	A	15	
License Center	A	25	
License Cat	A	5	
Issue Date	D		
Expdate	D		
Remarks	A		

**Table 4.7 Address**

Filed Name	Type	Size	Key
Code	A	5	*
Phone No1	A	15	
Phone No2	A	15	
Address1	A	15	
Address2	A	20	

**Table 4.8 Plane**

Filed Name	Type	Size	Key
Name	A	8	*
Performance	M	1	
Production	M	1	
Weight	M	1	

**Table 4.9 Maintenance**

Filed Name	Type	Size	Key
Plane Name	A	10	*
Day	A	10	
Starting Date	D		
Starting Time	T		
Finished Date	D		
Finished Time	T		
Manager	A	15	
Note	A	30	

**Table 4.10** Plane Details

Field Name	Type	Size	Key
planeName	A	10	*
planeType	A	10	
planeSize	S		
No_Passenger	N		
No_Crew	N		
No_Pilot	N		
No_FlightOfficer	N		
No_FlightEngineer	N		
No_Cabin	N		
No_SeniorCabin	N		
No_PartTimeHostes	N		

**Table 4.11** Password

Field Name	Type	Size	Key
Name	A	10	*
Username	A	10	
Password	A	10	

## 4. 2 Master/Detail Tables:

Setting up a master/detail relationship with the Delphi Table component is easy. Let me explain a master/detail relationship and then I'll show you how to set up one. Let's say you have a table called PLANE that contains information on your customers. That table will likely be indexed on a field called Name.

Let's further assume that you have a table called Maintenance that contains a list of all orders placed by your customers. Naturally, this table would also have a Plane Name field. Now let's say you want to browse the table containing all your customers. Wouldn't it be nice if you

Could see each customer's orders while you browse? A master/detail table enables you to do that



1. Start with a new application. Place a Table component on the form. Set its properties as follows:

Name	Master
DatabaseName	Flight
TableName	Plane.db

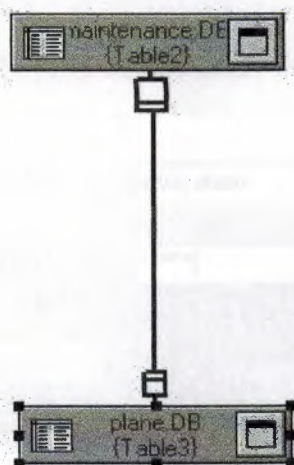
2. Place a DataSource component on the form and set its Dataset property to Master.
3. Now place a second Table component on the form and change its Name property to Details. You'll set the rest of this table's properties in just a minute.
4. Place a second DataSource component on the form. Change its DataSource property to Details.
5. Click on the Details Table component. Change its properties as follows:

DatabaseName	Flight
TableName	Maintenance.db
MasterSource	DataSource1

6. Click on the ellipsis button next to the Master Fields property. The Field Link Designer dialog box is displayed.
7. At the top of the Field Link Designer dialog box is a combo box labeled Available Indexes. Select the Name index from this combo box.
8. Now both the Detail Fields list box and the Master Fields list box have a Name entry. Select Name in each of these list boxes and click the Add button to create the relationship. The Joined Fields list box shows that the two tables are joined by their Name fields.
9. Click OK to close the Field Link Designer dialog boxes.
10. Drop two DBGrid components on the form and link one to DataSource1 and the other to DataSource2.

11. Change the Active property of both tables to True. The Master table will show all customers, and the Details table will show the orders for each customer.

What you just did was create a relationship between the master table and the detail table. This relationship joined these two tables through a common field: Name to fully understand what this means, run the program and move from record to record in the master table. As you select a customer name in the master table, you will see only that customer's orders in the detail table.



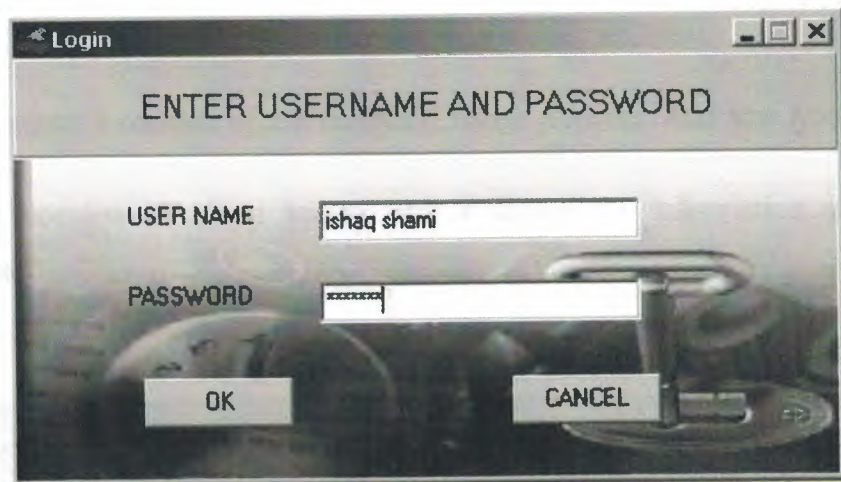
**Figure 4.1** Master/Detail Tables



## 4.3 Database Forms:

### 4.3.1 Log-in screen:

In order to protect our software a high level of security must be applied, so when the program runs it ask the operator to enter the user name and his/her own password to accomplish the iteration process. When the programs recognize the operator, the main menu screen will accrue. in case of unrecognizing alert message will occur tells that "invalid password, try again"



**Figure 4.1** Log-In Screens

### 4.3.2 Main Menu Screen:

- 1- File Button: we can use this button to Change program background And Exit from program.
- 2- Crew Information: The new information (personal Information ,Address ,License) should be applied through this button
- 3- Flight: It consists of five buttons each button has a specific mission ,and these missions as follow:

3.1- Add new flight.

3.2-Flight Details.

3.4-Show all.

3.5-Member Available.

4-Message: write new message.

5-Aircraft: It consists of tow buttons (Aircraft Information, Maintenance).

6- Search: The usage of this button is to find a certain data that we entered before about the crew member and aircraft.

7-Report

8-user Accounts: it consists of tow buttons (Change Account, Add new Account).

9-Help: it consists of about button and it shows the information about the programmer.



**Figure 4.2** Main Menu Screen



### 4.3.3 Member Information Screen:

The usage of this form is to apply information about new members like Name, Duty, Age, Nationality, Date of Birth, .....etc.

**Table 4.2.1** Button Task

Button	Task
ADD	Inserts a record in the dataset with the given field data and posts the edit
Delete	To Delete unnecessary inserted data
Post	Writes the edited record data to the database or to the cached update buffer
Next	Moves the cursor to the next record.
Prev	Moves the cursor to the previous record.
Last	Positions the cursor on the last record in the dataset.
First	Moves the cursor to the first record in the dataset
Browse	To Add New Picture
Close	Closes the dataset and Form.

The screenshot displays a software window titled "Member Information". At the top, there is a header bar with the title. Below the header, a row of buttons includes "First", "Prev", "Next", "Last", "ADD", "Delete", and "Post". On the left side, there is a "BROWSE" button and a placeholder for a member photo. The main area contains two columns of input fields: "Code" (with value "NVR"), "Name" (with value "AHMAD"), "Surname" (with value "ALI"), "Date OF Birth" (with value "10/12/1978"), "Place OF Birth" (with value "JAMMAN"), "Name OF Mother" (with value "MAEE"), "Number OF Passport" (with value "RG592874"), "Nationality" (with value "JORDANIAN"), "Duty" (with value "A310-PILOT"), "Married" (with radio buttons for "Yes" and "No"), and "Gender" (with radio buttons for "Male" and "Female").

**Figure 4.3** Member Information

#### 4.3.4 Address Screen:

This Screen allows us to enter the address of the crewmembers

The Address Screen displays a form for entering address information and a table of crew members. The form fields are:

- Code: AHM
- PhoneNo1: 0023536
- Address 1: DFFFFV
- PhoneNo2: 009025282554
- Address 2: DFFDDFWOV

The table below shows the list of crew members:

Code	Name	Surname	Date of birth	Place of birth	Name of mother	Sex	No of passp
AHM	AHMAD	ALI	10/12/1970	AMMAN	MAEE		FG558874
ALI	ali	fathi	10/14/1972	London	maha		KL254142
BAN	GULBAN	ERCAN	6/6/1980	NICOSIA	SDSCCD		DD15585
FAD	FADI	AALIA	6/4/1965	SAEDA	DFDVV		GG36554

Figure 4.4 Address Screen

#### 4.3.5 License Screen:

This screen allows us to apply general information about crewmembers license that are so essentially to accomplish the availability procedures.

The License Screen displays a form for entering license information and a table of crew members. The form fields are:

- Code: AHM
- License NO: HJ254485
- Issue Date: 10/8/2002
- License Center: HJ254485
- Exp Date: 5/6/2020
- License Category: A34
- Remarks: NEW

The table below shows the list of crew members:

Code	Name	Surname	Sex	No of passport	Nationality	Duty
AHM	AHMAD	ALI		FG558874	JORDANIAN	A310 PILOT
ALI	ali	fathi		KL254142	UK	cdkd
BAN	GULBAN	ERCAN		DD15585	CYPRUS	PART TIME HOSTE
FAD	FADI	AALIA		GG36554	LEBNON	B727 FLIGHT ENG
GUL	GULSUM	ALI		FV1828	CYPRUS	CABIN CHIEF

Figure 4.5 License Screen



### 4.3.6 Add new Flight Screen:

1-The user will be chosen the member from combo-box on the form the member who not Available will not appear in combo-box then the user add flight name, Plane type, date departure time, Arrival time.

2-The plane type should be chosen from the plane type combo-box.

3- To return back to the main menu click on the close button.

The screenshot shows a software interface for adding a new flight record. The window has a title bar 'Add new Record' and a main title 'ADD NEW FLIGHT'. A toolbar at the top includes buttons for 'SORT', 'ADD', 'FIRST', 'PREV', 'NEXT', 'LAST', 'DELETE', and a 'del' button. The interface is split into two main sections. The left section, 'Choose Cockpit', lists various roles with corresponding dropdown menus: Captain (AHMAD(A310)), Flight Officer (ALAA(A310)), Flight Engineer (MAHMOUD(A310) with a list of other names), Cabin Chief (MRAM), Senior Cabin (ARIZU), and Part time Hostes (FATEN, AYSHA). The right section contains dropdown menus for Plane type (A310), Flight NO (100), Departure Time (06:15), Arrival Time (17:30), Date (10/2/2002), and Route (DLM-MAN-DLM). A 'CLOSE' button is located at the bottom right.

Figure 4.6 Add new Flight Screen

### 4.3.7 Flight Calculation Screen:

This form usage to Calculate total of Flight time, duty time and rest time for Specific date e.g. For one month.

**Figure 4.7** Flight Calculation Screen

#### 4.3.8 View Crew Screen:

Flight	Date	DepTime	ArrTime	FTime	DTime	RTime	FlightNo
DLM-MAN-DLM	10/2/2002	06:15	17:30	11.15	12.45	22.3	100
GEC-IST-GEC	10/4/2002	15:15	20:45	5.3	7	10.6	150
GEC-ADB-STN-ADB-GEC	11/1/2002	06:40	20:50	14.1	15.4	28.2	200
ECN-IST-ECN	12/12/2002	05:00	08:00	3	4.3	8	250
ECN-ADB-ECN	2/26/2002	02:45	06:00	3.15	4.45	8	300
ECN-DLM-TLV-DLM-TLV-ECN	4/18/2002	06:00	23:00	17	18.3	34	350
GEC-ADB-LHR-ADB-GEC	11/18/2002	08:00	23:35	15.35	17.05	30.7	400

**Figure 4.8** View Crew Screen

#### 4.3.9 Members Available screen:

This form usage to show the user who member is available the user will be put the date and the code of members on the Edit box1 and Edit box2 then press ok.



FlightNo	Flight	Date	Dep Time	Arr Time	Plane	Plane	Plane	Plane	PlaneName
103	DLM-MAN-DLM	10/2/2002	06:15:00	17:30:00	11.15	22.3			12.45 A310
150	GEC-IST-GEC	10/4/2002	15:15:00	20:45:00	5.3	10.6			7 A310
200	GEC-ADB-STN-ADB-GEC	11/1/2002	06:40:00	20:50:00	14.1	26.2			15.4 B727
250	ECN-IST-ECN	12/12/2002	05:00:00	08:00:00	3	8			4.3 B737
300	ECN-ADB-ECN	2/26/2002	02:45:00	06:00:00	3.15	8			4.45 B727
350	ECN-DLM-TLV-DLM-TLV-ECN	4/18/2002	06:00:00	23:00:00	17	34			18.3 B737
400	GEC-ADB-LHR-ADB-GEC	11/18/2002	08:00:00	23:35:00	15.35	30.7			17.05 A310

**Figure 4.9 Members Available Screen**

#### 4.3.10 Aircraft screen:

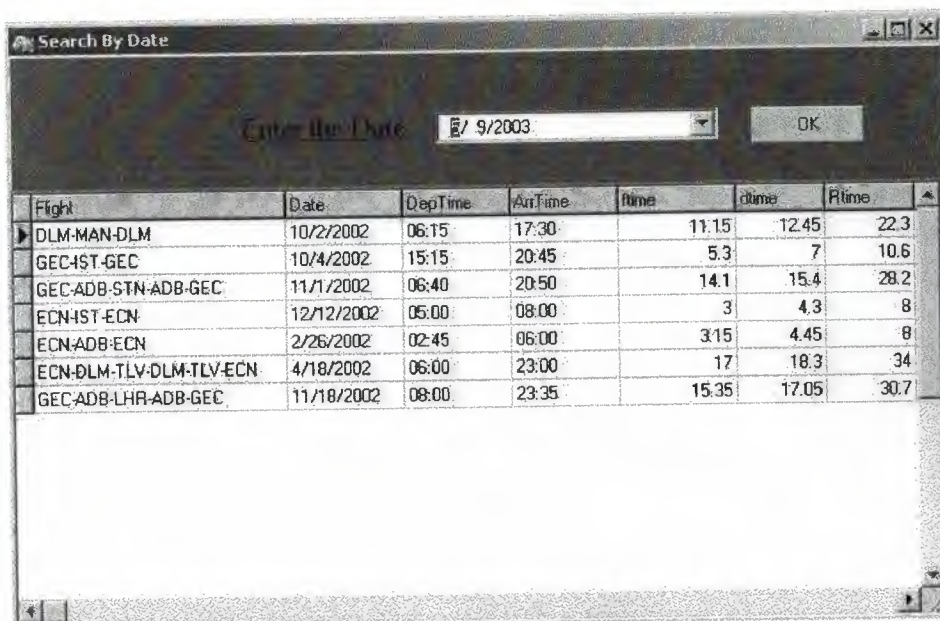
The usage of this form is to apply information about new aircraft that has been joined to the air wing of the company the specification of new aircraft is formed through this form.

Plan Type	<input type="text" value="jet"/>	NO OF Flight Officer	<input type="text" value="1"/>
Plan Name	<input type="text" value="TCJCO"/>	NO OF Flight Engineer	<input type="text" value="1"/>
Plane Size	<input type="text" value="246"/>	NO OF Cabin	<input type="text" value="2"/>
NO OF Passenger	<input type="text" value="100"/>	NO OF Senior Cabin	<input type="text" value="2"/>
NO OF Crew	<input type="text" value="3"/>	NO OF Part Hostes	<input type="text" value="3"/>
NO OF Pilot	<input type="text" value="2"/>		

**Figure 4.10 Aircraft Screen**

### 4.3.11 Search Screens:

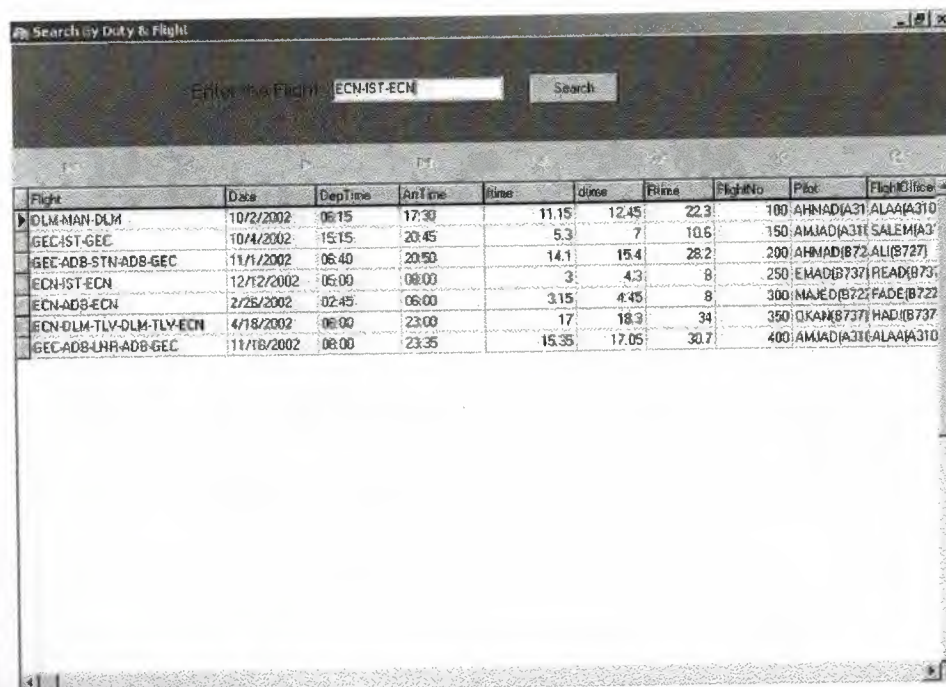
#### 1-by Date



Flight	Date	DepTime	ArrTime	time	dtime	Rtime
DLM-MAN-DLM	10/2/2002	06:15	17:30	11.15	12.45	22.3
GEC-IST-GEC	10/4/2002	15:15	20:45	5.3	7	10.6
GEC-ADB-STN-ADB-GEC	11/1/2002	06:40	20:50	14.1	15.4	28.2
ECN-IST-ECN	12/12/2002	05:00	08:00	3	4.3	8
ECN-ADB-ECN	2/26/2002	02:45	06:00	3.15	4.45	8
ECN-DLM-TLV-DLM-TLV-ECN	4/18/2002	06:00	23:00	17	18.3	34
GEC-ADB-LHR-ADB-GEC	11/18/2002	08:00	23:35	15.35	17.05	30.7

Figure 4.11 Searches by Date Screen

#### 2-By Flight:



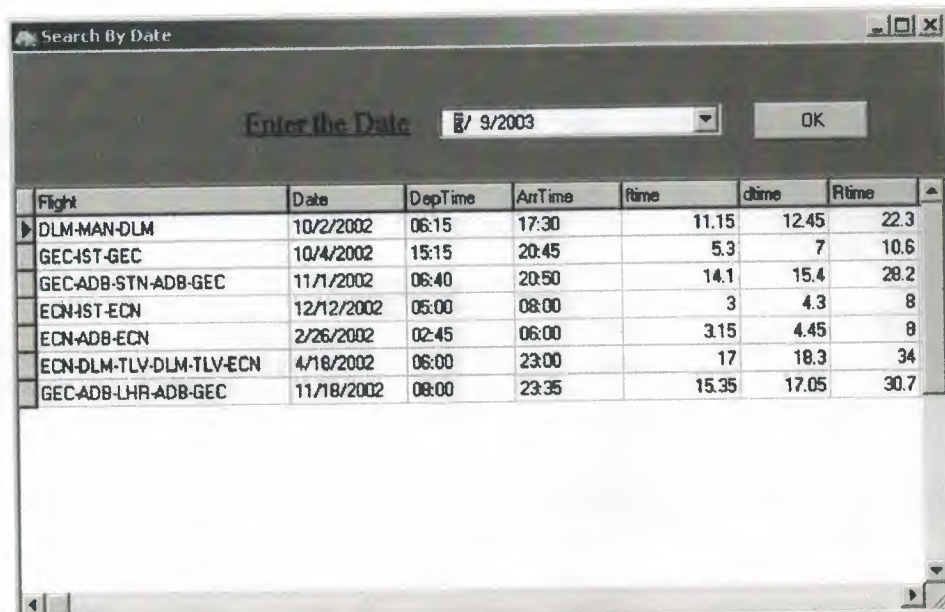
Flight	Date	DepTime	ArrTime	time	dtime	Rtime	FlightNo	Pilot	FlightOffice
DLM-MAN-DLM	10/2/2002	06:15	17:30	11.15	12.45	22.3	100	AHMAD(A31) ALAA(A310)	
GEC-IST-GEC	10/4/2002	15:15	20:45	5.3	7	10.6	150	AMJAD(A31) SALEM(A31)	
GEC-ADB-STN-ADB-GEC	11/1/2002	06:40	20:50	14.1	15.4	28.2	200	AHMAD(B72) ALI(B727)	
ECN-IST-ECN	12/12/2002	05:00	08:00	3	4.3	8	250	EMAD(B737) READ(B737)	
ECN-ADB-ECN	2/26/2002	02:45	06:00	3.15	4.45	8	300	MAJED(B727) FADE(B722)	
ECN-DLM-TLV-DLM-TLV-ECN	4/18/2002	06:00	23:00	17	18.3	34	350	OKAN(B737) HADI(B737)	
GEC-ADB-LHR-ADB-GEC	11/18/2002	08:00	23:35	15.35	17.05	30.7	400	AMJAD(A31) ALAA(A310)	

Figure 4.12 Searches by Flight Screen



### 4.3.11 Search Screens:

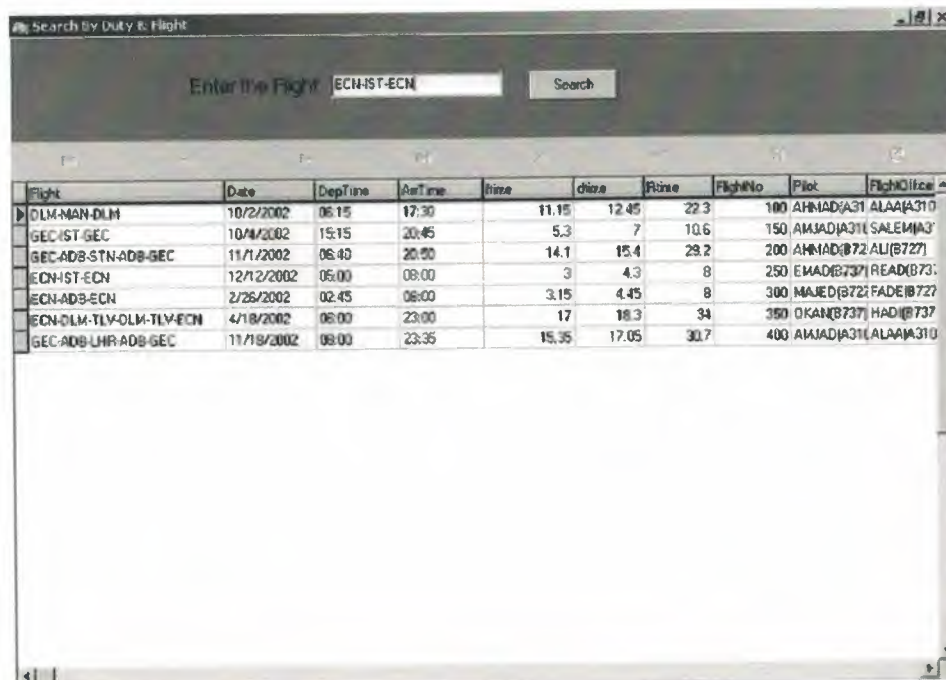
#### 1-By Date



Flight	Date	DepTime	ArrTime	time	dtime	Rtime
DLM-MAN-DLM	10/2/2002	06:15	17:30	11.15	12.45	22.3
GEC-IST-GEC	10/4/2002	15:15	20:45	5.3	7	10.6
GEC-ADB-STN-ADB-GEC	11/1/2002	06:40	20:50	14.1	15.4	29.2
ECN-IST-ECN	12/12/2002	05:00	08:00	3	4.3	8
ECN-ADB-ECN	2/26/2002	02:45	06:00	3.15	4.45	8
ECN-DLM-TLV-DLM-TLV-ECN	4/18/2002	06:00	23:00	17	18.3	34
GEC-ADB-LHR-ADB-GEC	11/18/2002	08:00	23:35	15.35	17.05	30.7

Figure 4.11 Searches by Date Screen

#### 2-By Flight:

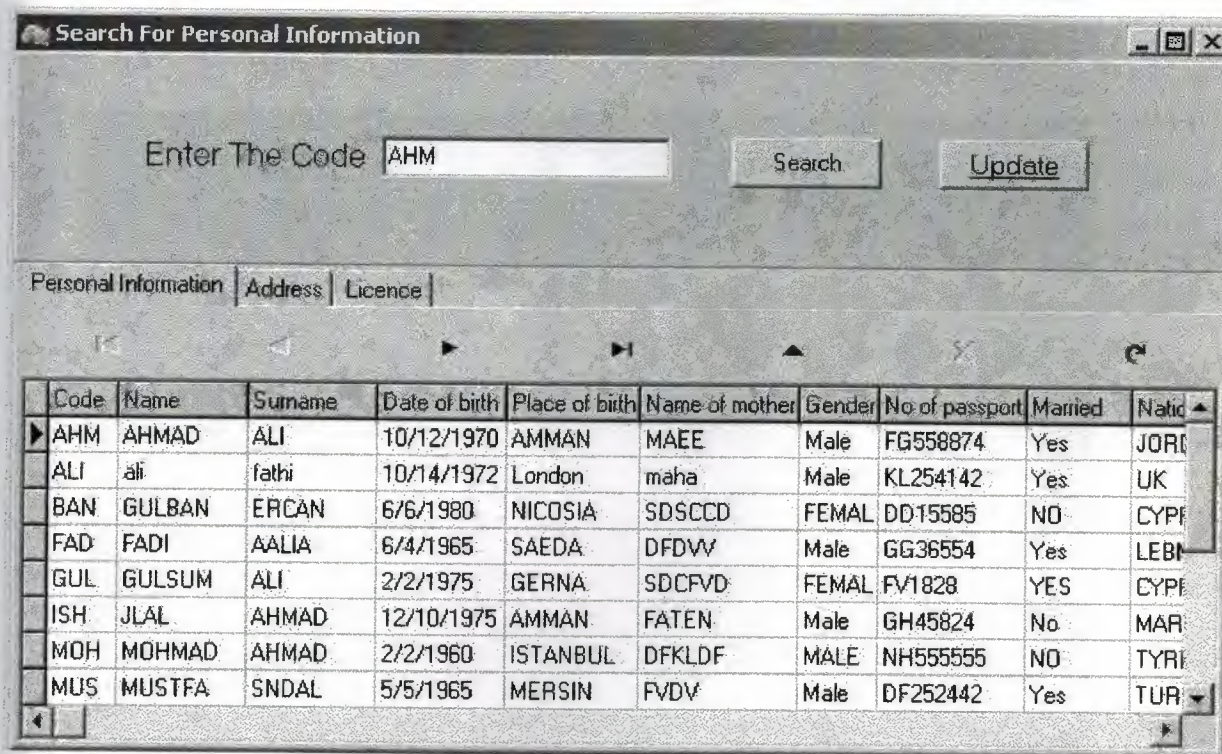


Flight	Date	DepTime	ArrTime	time	dtime	Rtime	FlightNo	Pilot	FlightIce
DLM-MAN-DLM	10/2/2002	06:15	17:30	11.15	12.45	22.3	100	AHMAD(A31 ALAA(A310	
GEC-IST-GEC	10/4/2002	15:15	20:45	5.3	7	10.6	150	AMJAD(A31(SALEM(A3	
GEC-ADB-STN-ADB-GEC	11/1/2002	06:40	20:50	14.1	15.4	29.2	200	AHMAD(B72 ALI(B727	
ECN-IST-ECN	12/12/2002	05:00	08:00	3	4.3	8	250	EMAD(B737) READ(B73	
ECN-ADB-ECN	2/26/2002	02:45	06:00	3.15	4.45	8	300	MAJED(B722) FADE(B727	
ECN-DLM-TLV-DLM-TLV-ECN	4/18/2002	06:00	23:00	17	18.3	34	350	OKAN(B737) HAD(B737	
GEC-ADB-LHR-ADB-GEC	11/18/2002	08:00	23:35	15.35	17.05	30.7	400	AMJAD(A31( ALAA(A310	

Figure 4.12 Searches by Flight Screen

### 3 Search for Members Information:

The usage of this form to search for personal information when the user put the code of members and press search button all the information about personal information , address and license will be appear in the Grid .



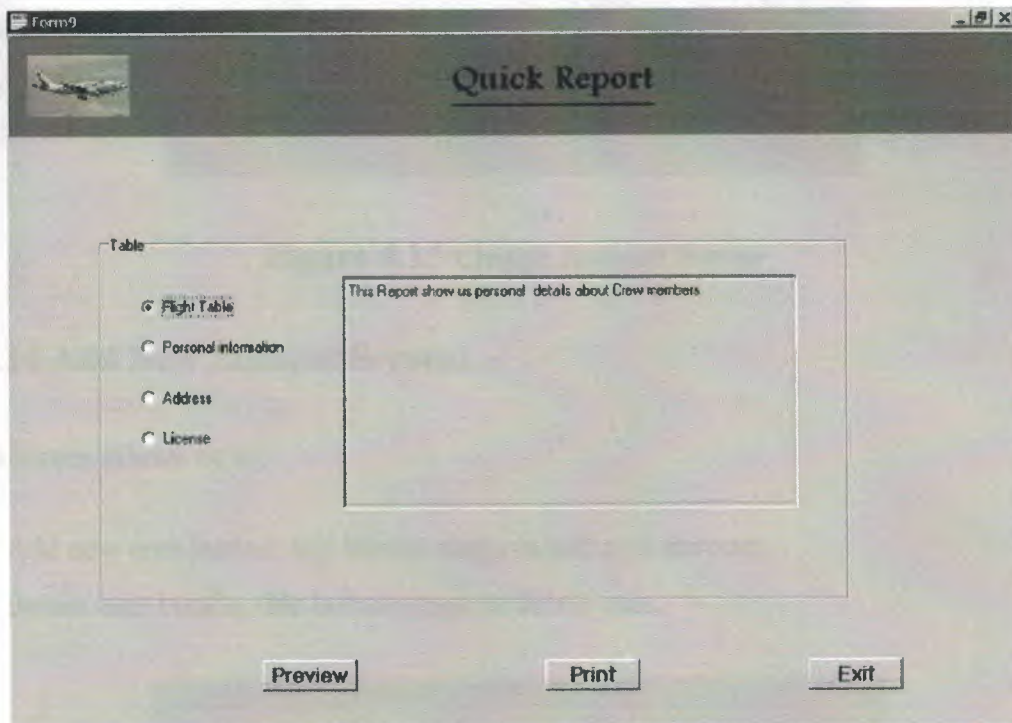
Code	Name	Surname	Date of birth	Place of birth	Name of mother	Gender	No. of passport	Married	Nationality
AHM	AHMAD	ALI	10/12/1970	AMMAN	MAEE	Male	FG558874	Yes	JORD
ALI	ali	fathi	10/14/1972	London	maha	Male	KL254142	Yes	UK
BAN	GULBAN	ERCAN	6/6/1980	NICOSIA	SDSCCD	FEMAL	DD15585	NO	CYPR
FAD	FADI	AALIA	6/4/1965	SAEDA	DFDVV	Male	GG36554	Yes	LEBN
GUL	GULSUM	ALI	2/2/1975	GERNA	SDCFVD	FEMAL	FV1828	YES	CYPR
ISH	JLAL	AHMAD	12/10/1975	AMMAN	FATEN	Male	GH45824	No	MAR
MOH	MOHMAD	AHMAD	2/2/1960	ISTANBUL	DFKLDF	MALE	NH555555	NO	TYRI
MUS	MUSTFA	SNDAL	5/5/1965	MERSIN	FVDV	Male	DF252442	Yes	TUR

**Figure 4.13** Search for Members Information Screen



#### 4.3.12 Report Screen:

A database program is not complete without some way of viewing and printing data, and that is where reports enter the picture. Up to this point, you have been looking at ways to view individual records this Screen allows us to view the Report and print out.



**Figure 4.14** Report Screen

#### 4.3.13 Change Account Screen:

this screen allows us to change user account when the user write his/her name the username and password will be appear in the screen automatically after this the user will be press change button to change the username and password

Finally, by pressing ok button the account will be update.

**Figure 4.15** Change Account Screen

#### 4.3.14 Add New Account Screen:

This screen allows us to:

- 1- Add new user button: this button usage to add new account.
- 2- Delete user button: this button usage to delete user.

**Figure 4.16** Add New Account Screen



## **CONCLUSION**

There are a lot of things in this project that I have learned for the very first time and even though not all the things I wanted to achieve or do in this project have actually appeared in this project but this is mainly because of the lack of time and knowledge in programming with Delphi, for example I wanted to link flight schedules to my Delphi program using Paradox, but could not do this due to complexity of aircraft schedule and also not to forget to mention that this consists of a huge information database which means more time in doing this. But we know how the saying goes "where there is a will there is a way". I have very high hopes in expanding the capability of this program in near future and from there I will take-off in mastering Delphi to design any program I want experience which is very important tool that I will need to tackle any obstacles being faced in the future.

## REFERENCES

- [1] Mastering Delphi 5, sybex, Marco Cantu.
- [2] Delphi 6, Ray, Dr.ramez qudseaha.
- [3] Teach Yourself Borland Delphi 4 in 21 Days, Macmillan Computer Publishing, Scotts Valley.
- [4] <http://www.about.com>.
- [5] <http://www.marcocantu.com>.
- [6] <http://www.DelphiMag.com>.
- [7]<http://www.airlines.net>

## APPENDIX

### PROGRAM CODE

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, jpeg, ExtCtrls, ComCtrls, Menus, DB, StdCtrls, Buttons,
  ImgList,
  ActnList, ExtDlgs, ToolWin;

type
  TAviation = class(TForm)
    TabControl1: TTabControl;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    Message1: TMenuItem;
    NewMessage1: TMenuItem;
    Search1: TMenuItem;
    View1: TMenuItem;
    Help1: TMenuItem;
    New1: TMenuItem;
    Repol: TMenuItem;
    QReport1: TMenuItem;
    Search2: TMenuItem;
    About1: TMenuItem;
    ImageList1: TImageList;
    ActionList1: TActionList;
    Action1: TAction;
    BitBtn1: TBitBtn;
    Image3: TImage;
```

```

RestTime1: TMenuItem;
new2: TMenuItem;
Address1: TMenuItem;
License1: TMenuItem;
MedicalReport1: TMenuItem;
raining1: TMenuItem;
FlightOperation1: TMenuItem;
AddNewFlight1: TMenuItem;
Showall1: TMenuItem;
Button1: TButton;
BitBtn3: TBitBtn;
BitBtn4: TBitBtn;
CalculationReport1: TMenuItem;
Image2: TImage;
HeaderControl1: THeaderControl;
ToolBar1: TToolBar;
ToolButton1: TToolButton;
ToolButton2: TToolButton;
ToolButton3: TToolButton;
ToolButton4: TToolButton;
ToolButton6: TToolButton;
ToolButton8: TToolButton;
ToolButton11: TToolButton;
Image1: TImage;
Label5: TLabel;
Label4: TLabel;
ToolButton12: TToolButton;
ToolButton13: TToolButton;
ByDate1: TMenuItem;
forSpecificDate1: TMenuItem;
ByDutyFlight1: TMenuItem;
PersonalInformation1: TMenuItem;
N1: TMenuItem;
membersAvilable1: TMenuItem;
Image4: TImage;

UserAccounts1: TMenuItem;
ChangeAccount1: TMenuItem;

```





```
CreatanewAccount1: TMenuItem;
FlightDetails1: TMenuItem;
AddNewFlight2: TMenuItem;
N2: TMenuItem;
FlightOperation2: TMenuItem;
FlightTime1: TMenuItem;
Maintenancel: TMenuItem;
SelectmembersAvilable1: TMenuItem;
procedure Exit1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure NewMessage1Click(Sender: TObject);
procedure New1Click(Sender: TObject);
procedure About1Click(Sender: TObject);
procedure Label1Click(Sender: TObject);
procedure Label2Click(Sender: TObject);
procedure QReport1Click(Sender: TObject);
procedure Label3Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Background1Click(Sender: TObject);
procedure RestTime1Click(Sender: TObject);
procedure Address1Click(Sender: TObject);
procedure License1Click(Sender: TObject);
procedure AddNewFlight1Click(Sender: TObject);
procedure Showall1Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);
procedure CalculationReport1Click(Sender: TObject);
procedure ToolButton1Click(Sender: TObject);
procedure ToolButton2Click(Sender: TObject);
procedure ToolButton3Click(Sender: TObject);
procedure ToolButton4Click(Sender: TObject);
procedure ByDate1Click(Sender: TObject);
procedure forSpecificDate1Click(Sender: TObject);
procedure ByDutyFlight1Click(Sender: TObject);
procedure PersonalInformation1Click(Sender: TObject);
procedure ToolButton7Click(Sender: TObject);
```

```

procedure ToolButton8Click(Sender: TObject);
procedure membersAvilable1Click(Sender: TObject);
procedure ToolButton11Click(Sender: TObject);
procedure ToolButton9Click(Sender: TObject);
procedure ChangeAccount1Click(Sender: TObject);
procedure CreatanewAccount1Click(Sender: TObject);
procedure ToolButton10Click(Sender: TObject);
procedure AddNewFlight2Click(Sender: TObject);
procedure showFlight1Click(Sender: TObject);
procedure FlightOperation2Click(Sender: TObject);
procedure FlightTime1Click(Sender: TObject);
procedure MaintenancelClick(Sender: TObject);
procedure ToolButton6Click(Sender: TObject);
procedure SelectmembersAvilable1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
  Aviation: TAviation;

implementation

uses Unit2,unit3,unit4,unit5,unit6,unit7, Unit9, Unit8, Unit11, Unit12,
  Unit13,unit14, Unit15, Unit16, Unit18, Unit19, Unit21, Unit20, Unit22,
  Unit17, Unit23, Unit25, Unit26, Unit24, Unit27,unit28, Unit30, Unit31,
  Unit33, Unit34;

{$R *.dfm}

function IsPrime (N: LongInt): Boolean;
var
  Test: LongInt;
begin
  IsPrime := True;
  for Test := 2 to N - 1 do
    begin

```

```

    if (N mod Test) = 0 then
    begin
        IsPrime := False;
        break; {jump out of the for loop}
    end;
    end;
    end;
    end;
procedure TAviation.Exit1Click(Sender: TObject);
begin
    close;
    form28.Close;
end;

procedure TAviation.Button2Click(Sender: TObject);
begin
    form3.ShowModal;
end;

procedure TAviation.NewMessage1Click(Sender: TObject);
begin
    form5.ShowModal;
end;

procedure TAviation.New1Click(Sender: TObject);
begin
    aircraft.ShowModal;
end;

procedure TAviation.About1Click(Sender: TObject);
begin
    form7.ShowModal;
end;

procedure TAviation.Label1Click(Sender: TObject);
begin
    flight.ShowModal;

end;

```



```

procedure TAviation.Label2Click(Sender: TObject);
begin
    form3.ShowModal;
end;

procedure TAviation.QReport1Click(Sender: TObject);
begin
    form9.ShowModal;
end;

procedure TAviation.Label3Click(Sender: TObject);
begin
    Aircraft.ShowModal;
end;

procedure TAviation.BitBtn1Click(Sender: TObject);
begin
    form11.ShowModal;
end;

procedure TAviation.FormCreate(Sender: TObject);
var
    I: Integer;
    SplashAbout: TForm13;
begin
    // create and show the splash form
    SplashAbout := TForm13.Create (Application);
    SplashAbout.MakeSplash;
    // standard code...
    for I := 1 to 4000 do
        if IsPrime (I) then
            {ListBox1.Items.Add (IntToStr (I));}
    // get rid of the splash form, after a while
    SplashAbout.Timer1.Enabled := True;

    label4.Caption:=datetostr(date);

```

```

label5.Caption:=timetostr(time);

end;

procedure TAviation.Button1Click(Sender: TObject);
begin
form24.show;
//flight.Show;
//form15.show;
end;

procedure TAviation.BackGround1Click(Sender: TObject);
var
str:string;
begin

if OpenPictureDialog1.Execute then
begin
str:=OpenPictureDialog1.FileName;
Image4.Picture.LoadFromFile(str);
image4.Stretch:=true;

Image1.Picture.LoadFromFile(str);
image1.Stretch:=true;

end;

end;

procedure TAviation.RestTime1Click(Sender: TObject);
begin
form19.ShowModal;
end;

procedure TAviation.Address1Click(Sender: TObject);
begin

```

```

form21.show;
end;

procedure TAviation.License1Click(Sender: TObject);
begin
form20.show;
end;

procedure TAviation.AddNewFlight1Click(Sender: TObject);
begin
form18.ShowModal;
end;

procedure TAviation.Showall1Click(Sender: TObject);
begin

form22.show;
end;

procedure TAviation.BitBtn3Click(Sender: TObject);
begin
flight.ShowModal;
end;

procedure TAviation.BitBtn4Click(Sender: TObject);
begin
form12.ShowModal;

end;

procedure TAviation.CalculationReport1Click(Sender: TObject);
begin
form26.ShowModal;
end;

procedure TAviation.ToolButton1Click(Sender: TObject);
begin
form11.ShowModal;

```



end;

procedure TAviation.ToolButton2Click(Sender: TObject);

begin

form3.ShowModal;

end;

procedure TAviation.ToolButton3Click(Sender: TObject);

begin

form14.ShowModal;

end;

procedure TAviation.ToolButton4Click(Sender: TObject);

begin

form24.show;

end;

procedure TAviation.ByDate1Click(Sender: TObject);

begin

form15.ShowModal;

end;

procedure TAviation.forSpecificDate1Click(Sender: TObject);

begin

form17.ShowModal;

end;

procedure TAviation.ByDutyFlight1Click(Sender: TObject);

begin

form23.ShowModal;

end;

procedure TAviation.PersonalInformation1Click(Sender: TObject);

begin

form24.ShowModal;

end;

procedure TAviation.ToolButton7Click(Sender: TObject);

```

begin
form3.ShowModal;
end;

procedure TAviation.ToolButton8Click(Sender: TObject);
begin
form30.ShowModal;
end;

procedure TAviation.membersAvilable1Click(Sender: TObject);
begin
form27.ShowModal;
end;

procedure TAviation.ToolButton11Click(Sender: TObject);
begin
Close;
end;

procedure TAviation.ToolButton9Click(Sender: TObject);
begin
FORM16.Show;
end;

procedure TAviation.ChangeAccount1Click(Sender: TObject);
begin
form30.ShowModal;

end;

procedure TAviation.CreatanewAccount1Click(Sender: TObject);
begin
form34.ShowModal;
end;

procedure TAviation.ToolButton10Click(Sender: TObject);
begin
form13.Show ;

```

end;

procedure TAviation.AddNewFlight2Click(Sender: TObject);

begin

flight.ShowModal;

end;

procedure TAviation.showFlight1Click(Sender: TObject);

begin

form13.ShowModal;

end;

procedure TAviation.FlightOperation2Click(Sender: TObject);

begin

form14.ShowModal;

end;

procedure TAviation.FlightTime1Click(Sender: TObject);

begin

form12.Show;

end;

procedure TAviation.MaintenancelClick(Sender: TObject);

begin

    maint.ShowModal;

end;

procedure TAviation.ToolButton6Click(Sender: TObject);

    var

    str:string;

begin

    if OpenPictureDialog1.Execute then

    begin

        str:=OpenPictureDialog1.FileName;

        Image4.Picture.LoadFromFile(str);

        image4.Stretch:=true;



```
Image1.Picture.LoadFromFile(str);  
image1.Stretch:=true;
```

```
end;
```

```
end;
```

```
procedure TAviation.SelectmembersAvilable1Click(Sender: TObject);
```

```
begin
```

```
form33.ShowModal;
```

```
end;
```

```
end.
```

```
unit Unit2;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
  Forms,
```

```
  Dialogs, StdCtrls, Mask, DBCtrls, DB, DBTables, ActnList, ImgList,  
  ComCtrls, ToolWin, DBActns, jpeg, ExtCtrls, Grids, DBGrids, ADODB;
```

```
type
```

```
  Tflight = class(TForm)
```

```
    DBGrid1: TDBGrid;
```

```
    Panel1: TPanel;
```

```
    Label1: TLabel;
```

```
    Label2: TLabel;
```

```
    Label3: TLabel;
```

```
    Label9: TLabel;
```

```
    DBEdit3: TDBEdit;
```

```
    ToolBar1: TToolBar;
```

```
    ToolButton2: TToolButton;
```

```
    ToolButton3: TToolButton;
```

```
    ToolButton4: TToolButton;
```

```
    ToolButton1: TToolButton;
```

```
    ToolBar2: TToolBar;
```

```
    ToolButton5: TToolButton;
```

```
    ToolButton6: TToolButton;
```

```
    DBEdit9: TDBEdit;
```

```
    ImageList1: TImageList;
```

```
    ActionList1: TActionList;
```

```
    DataSetFirst1: TDataSetFirst;
```

```
    DataSetPrior1: TDataSetPrior;
```

```
    DataSetLast1: TDataSetLast;
```

```
    DataSetNext1: TDataSetNext;
```

```
    DataSetInsert1: TDataSetInsert;
```

```
    DataSetDelete1: TDataSetDelete;
```

```
    DataSetInsert2: TDataSetInsert;
```

```
    Table2: TTable;
```

```

DataSource2: TDataSource;
Table2FlightCode: TStringField;
Table2ID: TStringField;
Table2Date: TDateField;
Table2Day: TStringField;
Table2Dep_Air: TStringField;
Table2Dep_Time: TTimeField;
Table2ArrAir: TStringField;
Table2ArrTime: TTimeField;
Table1: TTable;
DataSource1: TDataSource;
Table1PlaneName: TStringField;
Table1PlaneType: TStringField;
Table1Planesize: TSmallintField;
Table1NO_passenger: TFloatField;
Table1NO_Crew: TFloatField;
Table1NO_Captain: TFloatField;
Table1NO_Firstofficer: TFloatField;
Table1NO_FlightEngineer: TFloatField;
Table1NO_Cabin: TFloatField;
Table1NO_SeniorCabin: TFloatField;
Table1NO_PartHostes: TFloatField;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
procedure DataSource1DataChange(Sender: TObject; Field: TField);
procedure MonthCalendar1Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    flight: Tflight;

implementation

uses Unit3,unit14;

```



```
{ $R *.dfm }
```

```
procedure Tflight.DataSource1DataChange(Sender: TObject; Field: TField);  
begin  
  {monthCalendar1.date:=table1date.Value;}  
end;
```

```
procedure Tflight.MonthCalendar1Click(Sender: TObject);  
begin  
  table1.Edit;  
end;
```

```
end.
```

```
unit Unit3;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
  Forms,
```

```
  Dialogs, ComCtrls, ExtCtrls, DBCtrls, DB, DBTables, StdCtrls, Mask;
```

```
type
```

```
  TForm3 = class(TForm)
```

```
    PageControl1: TPageControl;
```

```
    TabSheet2: TTabSheet;
```

```
    TabSheet3: TTabSheet;
```

```
    DataSource1: TDataSource;
```

```
    Table1: TTable;
```

```
    DBEdit4: TDBEdit;
```

```
    DBEdit5: TDBEdit;
```

```
    DBEdit6: TDBEdit;
```

```
    Label4: TLabel;
```

```
    Label5: TLabel;
```

```
    Label6: TLabel;
```

```
    Label7: TLabel;
```

```
    Label8: TLabel;
```

```
    DBNavigator2: TDBNavigator;
```

```
    DataSource2: TDataSource;
```

```
    Button1: TButton;
```

```
    Label9: TLabel;
```

```
    Table1PlaneName: TStringField;
```

```
    Table1PlaneType: TStringField;
```

```
    Table1Planesize: TSmallintField;
```

```
    Label10: TLabel;
```

```
    Label11: TLabel;
```

```
    DBEdit9: TDBEdit;
```

```
    DBEdit10: TDBEdit;
```

```
    DBEdit11: TDBEdit;
```

```
    Label13: TLabel;
```

```

Table2: TTable;
DBNavigator3: TDBNavigator;
Label12: TLabel;
Table3: TTable;
DataSource3: TDataSource;
DBEdit14: TDBEdit;
DBEdit15: TDBEdit;
DBComboBox1: TDBComboBox;
DBComboBox2: TDBComboBox;
DBComboBox3: TDBComboBox;
DBComboBox4: TDBComboBox;
Table3FlightCode: TStringField;
Table3PlaneName: TStringField;
Table3Country1: TStringField;
Table3Available: TStringField;
Table3Country2: TStringField;
Table3Available2: TStringField;
Table3Country3: TStringField;
Table3Available3: TStringField;
Table2CapacityOfAir: TStringField;
Table2Nplane: TStringField;
Table2Country: TStringField;
Table2AirCode: TStringField;
Table2Available: TStringField;
procedure Button1Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form3: TForm3;

implementation

uses Unit4, unit2;

```

```
{ $R *.dfm }
```

```
procedure TForm3.Button1Click(Sender: TObject);
```

```
begin
```

```
form4.show;
```

```
end;
```

```
end.
```



```

unit Unit4;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, StdCtrls;

type
  TForm4 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form4: TForm4;

implementation

uses Unit3, unit1;

{$R *.dfm}

procedure TForm4.Button1Click(Sender: TObject);
var
  ch:string;
begin

```

```
form3.Label9.Caption:=edit1.Text;
```

```
edit1.Clear;
```

```
form4.Close;
```

```
end;
```

```
procedure TForm4.Button2Click(Sender: TObject);
```

```
begin
```

```
edit1.Clear;
```

```
form4.Close;
```

```
end;
```

```
end.
```

```
unit Unit5;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, ExtCtrls, DBCtrls, DB, DBTables, StdCtrls, Mask;
```

```
type
```

```
  TForm5 = class(TForm)
```

```
    DBEdit1: TDBEdit;
```

```
    DBEdit2: TDBEdit;
```

```
    DBMemo1: TDBMemo;
```

```
    DBMemo2: TDBMemo;
```

```
    Label1: TLabel;
```

```
    Label2: TLabel;
```

```
    Label3: TLabel;
```

```
    Label4: TLabel;
```

```
    Label5: TLabel;
```

```
    Table1: TTable;
```

```
    DataSource1: TDataSource;
```

```
    DBNavigator1: TDBNavigator;
```

```
    DBEdit3: TDBEdit;
```

```
  private
```

```
    { Private declarations }
```

```
  public
```

```
    { Public declarations }
```

```
end;
```

```
var
```

```
  Form5: TForm5;
```

```
implementation
```

```
  {$R *.dfm}
```

```
end.
```

```
unit Unit6;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
  Forms,  
  Dialogs, DB, DBTables, StdCtrls, Mask, DBCtrls, ExtCtrls, Grids,  
  DBGrids;
```

```
type
```

```
  Tmaint = class(TForm)
```

```
    Table1: TTable;
```

```
    Table2: TTable;
```

```
    Table3: TTable;
```

```
    DataSource1: TDataSource;
```

```
    DataSource2: TDataSource;
```

```
    DataSource3: TDataSource;
```

```
    DBMemo1: TDBMemo;
```

```
    DBMemo2: TDBMemo;
```

```
    DBMemo3: TDBMemo;
```

```
    DBEdit1: TDBEdit;
```

```
    DBEdit4: TDBEdit;
```

```
    DBEdit5: TDBEdit;
```

```
    DBEdit6: TDBEdit;
```

```
    DBEdit7: TDBEdit;
```

```
    DBEdit8: TDBEdit;
```

```
    DBEdit9: TDBEdit;
```

```
    DBEdit10: TDBEdit;
```

```
    Label1: TLabel;
```

```
    Label2: TLabel;
```

```
    Label3: TLabel;
```

```
    Label4: TLabel;
```

```
    Label5: TLabel;
```

```
    Label6: TLabel;
```

```
    Label7: TLabel;
```

```
    Label8: TLabel;
```

```
    Label9: TLabel;
```



```

Label10: TLabel;
Label11: TLabel;
Button1: TButton;
Button2: TButton;
Button3: TButton;
Button4: TButton;
Button5: TButton;
Button6: TButton;
Button7: TButton;
DBGGrid1: TDBGGrid;
Table3PlaneName: TStringField;
Table3PlaneType: TStringField;
Table3Planesize: TSmallintField;
Table3NO_passenger: TFloatField;
Table3NO_Crew: TFloatField;
Table3NO_Captain: TFloatField;
Table3NO_Firstofficer: TFloatField;
Table3NO_FlightEngineer: TFloatField;
Table3NO_Cabin: TFloatField;
Table3NO_SeniorCabin: TFloatField;
Table3NO_PartHostes: TFloatField;
Table2PlaneName: TStringField;
Table2Days: TStringField;
Table2StartingDate: TDateField;
Table2StartingTime: TTimeField;
Table2FinishedDate: TDateField;
Table2FinishedTime: TTimeField;
Table2Manager: TStringField;
Table2NOTE: TStringField;
Panell1: TPanel;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
private

```

```

    { Private declarations }
public
    { Public declarations }
end;

var
    maint: Tmaint;

implementation

uses Unit8;

{$R *.dfm}

procedure Tmaint.Button1Click(Sender: TObject);
begin
    table2.First;
end;

procedure Tmaint.Button2Click(Sender: TObject);
begin
    table2.Prior;
end;

procedure Tmaint.Button3Click(Sender: TObject);
begin
    table2.Next;
end;

procedure Tmaint.Button4Click(Sender: TObject);
begin
    table2.Last;
end;

procedure Tmaint.Button7Click(Sender: TObject);
begin
    table2.Insert;
end;

```

```
procedure Tmaint.Button6Click(Sender: TObject);  
begin  
table2.Delete;  
end;
```

```
procedure Tmaint.Button5Click(Sender: TObject);  
begin  
close;  
end;
```

```
end.
```

```

unit Unit7;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TForm7 = class(TForm)
    Image1: TImage;
    Image2: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label9: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form7: TForm7;

implementation

{$R *.dfm}

end.

```



```
unit Unit8;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
  Forms,
```

```
  Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, DB, ADODB, ActnList,  
  DBTables, StdCtrls, Mask, DBActns;
```

```
type
```

```
  TAircraft = class(TForm)
```

```
    Table1: TTable;
```

```
    DataSource1: TDataSource;
```

```
    Table1PlaneName: TStringField;
```

```
    Table1PlaneType: TStringField;
```

```
    Table1Planesize: TSmallintField;
```

```
    Table1NO_passenger: TFloatField;
```

```
    Table1NO_Crew: TFloatField;
```

```
    Table1NO_Captain: TFloatField;
```

```
    Table1NO_Firstofficer: TFloatField;
```

```
    Table1NO_FlightEngineer: TFloatField;
```

```
    Table1NO_Cabin: TFloatField;
```

```
    Table1NO_SeniorCabin: TFloatField;
```

```
    Table1NO_PartHostes: TFloatField;
```

```
    DBEdit1: TDBEdit;
```

```
    DBEdit2: TDBEdit;
```

```
    DBEdit3: TDBEdit;
```

```
    DBEdit4: TDBEdit;
```

```
    DBEdit5: TDBEdit;
```

```
    DBEdit6: TDBEdit;
```

```
    DBEdit7: TDBEdit;
```

```
    DBEdit8: TDBEdit;
```

```
    DBEdit9: TDBEdit;
```

```
    DBEdit10: TDBEdit;
```

```
    DBEdit11: TDBEdit;
```

```
    Label1: TLabel;
```

```
    Label2: TLabel;
```

```

Label13: TLabel;
Label14: TLabel;
Label15: TLabel;
Label16: TLabel;
Label17: TLabel;
Label18: TLabel;
Label19: TLabel;
Label110: TLabel;
Label111: TLabel;
Button1: TButton;
Button2: TButton;
Button3: TButton;
Button4: TButton;
Button5: TButton;
Panel1: TPanel;
Button6: TButton;
Button7: TButton;
Button8: TButton;
ActionList1: TActionList;
DataSetPost1: TDataSetPost;
procedure Button4Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure Button8Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Aircraft: TAircraft;

implementation

```

```
{ $R *.dfm }
```

```
procedure TAircraft.Button4Click(Sender: TObject);  
begin  
  table1.Last;  
end;
```

```
procedure TAircraft.Button2Click(Sender: TObject);  
begin  
  table1.Prior;  
end;
```

```
procedure TAircraft.Button1Click(Sender: TObject);  
begin  
  table1.First;  
end;
```

```
procedure TAircraft.Button3Click(Sender: TObject);  
begin  
  table1.Next;  
end;
```

```
procedure TAircraft.Button5Click(Sender: TObject);  
begin  
  close;  
end;
```

```
procedure TAircraft.Button6Click(Sender: TObject);  
begin  
  TABLE1.Insert;  
end;
```

```
procedure TAircraft.Button7Click(Sender: TObject);  
begin  
  if MessageDlg('Are you sure Want Delete this Record. Exit now?',  
    mtConfirmation, [mbYes, mbNo], 0) = mrYes then  
    table1.Delete;
```

end;

procedure TAircraft.Button8Click(Sender: TObject);

begin

{TABLE1.Append;

table1.Post;}

end;

end.



```

unit Unit11;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, StdCtrls, comobj;

type
  TForm11 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form11: TForm11;

implementation

  uses unit1;

  {$R *.dfm}

  procedure TForm11.Button1Click(Sender: TObject);
  const
    olmailitem=0;

```

```

var
o,m:variant;
begin
o:=createoleobject('outlook.application');
m:=o.createitem(olmailitem);
m.to:=edit1.Text;
m.subject:=edit2.Text;
m.body:=memol.Text;
m.send;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
close;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
edit1.Clear;
edit2.Clear;
memol.Clear;
end;

end.

```

```

unit Unit12;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, StdCtrls, Grids, DBGrids, Mask, DBCtrls, DB, DBTables,
  ExtCtrls;

type
  TForm12 = class(TForm)
    Table2: TTable;
    DataSource2: TDataSource;
    DBNavigator1: TDBNavigator;
    Panel1: TPanel;
    DBGrid1: TDBGrid;
    Panel2: TPanel;
    Label1: TLabel;
    Edit1: TEdit;
    Button2: TButton;
    Memo1: TMemo;
    Button1: TButton;
    Table2FlightCode: TStringField;
    Table2ID: TStringField;
    Table2Date: TDateField;
    Table2Day: TStringField;
    Table2Dep_Air: TStringField;
    Table2Dep_Time: TTimeField;
    Table2ArrAir: TStringField;
    Table2ArrTime: TTimeField;
    Table2flighttime: TTimeField;
    procedure Button1Click(Sender: TObject);
    procedure Table2CalcFields(DataSet: TDataSet);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  end;

```

```

    procedure Table2PlaneChange(Sender: TField);
    procedure Button5Click(Sender: TObject);

private

    { Private declarations }
public
    d,h,total,duty:real; { Public declarations }
    end;

var
    Form12: TForm12;

implementation

{$R *.dfm}

procedure TForm12.Button1Click(Sender: TObject);
begin
    Table2.Filter:='ID='''+edit1.text+'''; //code='B'
    Table2.Filtered:=true;

end;

procedure TForm12.Button2Click(Sender: TObject);

var
    // total:ttime ;
    bookmark:tbookmarkstr;
    g:string;
begin
    Bookmark := Table2.Bookmark;
    Table2.DisableControls;
    Total := 0;
    try
        Table2.First;

```



```

while not Table2.EOF do
begin
    Total := Total + Table2flighttime.Value;
    Table2.Next;
end;
finally
    // go back to the bookmark
    Table2.Bookmark := Bookmark;
    Table2.EnableControls;
end;
{MessageDlg ('Sum of new salaries is ' +
    Format ('%m', [Total]), mtInformation, [mbOk], 0); }
g:=timetostr(total);
memol.Lines.Text:=g;

end;

procedure TForm12.Table2CalcFields(DataSet: TDataSet);
begin
    table2flighttime.value:=table2arrtime.value-table2dep_time.value;
end;

procedure TForm12.Button3Click(Sender: TObject);
var
    duty,j,d,m:tttime;
    l:string;
begin
    table2.Last;
    j:=encodetime(00,30,00,00);
    d:=table2arrtime.value+j;

    m:=encodetime(1,00,00,00);
    table2.first;
    h:=table2dep_time.Value-m;

```

```

    duty:=d-h;
    l:=timetostr(duty);
    memo1.lines.text:=l;

end;

procedure TForm12.FormCreate(Sender: TObject);
var
d,j,m:ttime;

begin
    m:=encodetime(1,00,00,00);
    table2.first;
    h:=table2dep_time.Value-m;
end;

procedure TForm12.Button4Click(Sender: TObject);
var
max,ft,m,ftime,restT:ttime;
str:string;
begin
    form12.Button2Click(sender);
    m:=encodetime(08,00,00,10);

    fTime:=total+total;
    // showmessage(timetostr(ftime));

    max:=ftime;
    if m > max then
        max:=m;
    if duty > max then
        max:=duty;
    restT:=max;
    str:=timetostr(restT);
    memo1.lines.text:=str;

```

end;

procedure TForm12.Table2PlaneChange(Sender: TField);

begin

    showmessage('aaaa');

end;

procedure TForm12.Button5Click(Sender: TObject);

VAR

    STR:STRING;

begin

    //TABLE2.Next;

    //str:=Table2Plane.Value ;

    // table5.Filter:='plane='+str;

    //table5.Filtered:=true;

end;

end.

```

unit Unit13;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, jpeg;

type
  TForm13 = class(TForm)
    Image1: TImage;
    Timer1: TTimer;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    procedure MakeSplash;
  { Public declarations }
  end;

var
  Form13: TForm13;

implementation

{$R *.dfm}

procedure tform13.MakeSplash;

```



```
begin
BorderStyle := bsNone;
  BitBtn1.Visible := False;

  Show;
  Update;
end;
procedure TForm13.Timer1Timer(Sender: TObject);
begin
  Close;
  Release;
end;

end.
```

```

unit Unit14;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, Mask, DBCtrls, StdCtrls, Grids, DBGrids, DB, DBTables,
  ExtCtrls,
  ComCtrls;

type
  TForm14 = class(TForm)
    Panel1: TPanel;
    Edit1: TEdit;
    Button1: TButton;
    Label1: TLabel;
    Label5: TLabel;
    DBGrid1: TDBGrid;
    Panel2: TPanel;
    Memo1: TMemo;
    Button2: TButton;
    Memo2: TMemo;
    Label7: TLabel;
    Label8: TLabel;
    DateTimePicker1: TDateTimePicker;
    DateTimePicker2: TDateTimePicker;
    Memo3: TMemo;
    Label9: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

```

```

end;

var
    Form14: TForm14;

implementation
    uses unit2, unit15, Unit17;
{$R *.dfm}

procedure TForm14.Button1Click(Sender: TObject);
var
    datel, date2, str, str2: string;
begin
    {
        form17.Query1.Filter:='Pilot='''+edit1.text+''''or
        ''FlightEngineer='''+edit1.text+''''; // and Date='''+edit2.Text+'''';
        form17.Query1.Filtered:=true;
        edit1.Clear; }

    datel:=Datetostr(DateTimePicker1.Date);
    date2:=datetostr(DateTimePicker2.Date);
    str:='select * from calcpilot where calcpilot."Date" between
    '''+datel+''''And '''+date2+''''And calcpilot.pilot='''+edit1.text+''''or
    calcpilot.FlightEngineer='''+edit1.Text+''''or
    calcpilot.Flightofficer='''+edit1.Text+''''or
    calcpilot.CabinChief='''+edit1.Text+''''or
    calcpilot.SeniorCabin1='''+edit1.Text+''''or
    calcpilot.SeniorCabin2='''+edit1.Text+''''or
    calcpilot.PartTimeHostes1='''+edit1.Text+''''or
    calcpilot.partTimehostes2='''+edit1.Text+'''''''' ' ;
    form17.query1.SQL.Clear;
    form17.query1.SQL.Add(str);

    form17.query1.Open;

```

end;

procedure TForm14.Button2Click(Sender: TObject);

var

total2,total3, total:real ;

bookmark:tbookmarkstr;

t1,t2,t3:string;

begin

Bookmark := form17.Query1.Bookmark;

form17.Query1.DisableControls;

Total := 0;

total2:=0;

total3:=0;

try

form17.Query1.First;

while not form17.Query1.EOF do

begin

Total := Total + form17.Query1.ftime.Value;

total2:=total2+form17.Query1.dtime.Value;

total3:=total3+form17.Query1.Rtime.Value;

form17.Query1.Next;

end;

finally

// go back to the bookmark

form17.Query1.Bookmark := Bookmark;

form17.Query1.EnableControls;

end;

{MessageDlg ('Sum of new salaries is ' +

Format ('%m', [Total]), mtInformation, [mbOk], 0); }

t1:=floattostr(total);

memol.Lines.Text:=t1;



```

t2:=floattostr(total2);
memo2.Lines.Text:=t2;

t3:=floattostr(total3);
memo3.Lines.Text:=t3;

end;

procedure TForm14.Button3Click(Sender: TObject);
var
str2:string;
begin
//str2:='SELECT      *      from      calcpilot      where
calcpilot.pilot='''+edit1.text+'''+or
calcpilot.FlightEngineer='''+edit1.Text+'''+or
calcpilot.Flightofficer='''+edit1.Text+'''+or
calcpilot.CabinChief='''+edit1.Text+'''+or
calcpilot.SeniorCabin1='''+edit1.Text+'''+or
calcpilot.SeniorCabin2='''+edit1.Text+'''+or
calcpilot.PartTimeHostes1='''+edit1.Text+'''+or
calcpilot.partTimehostes2='''+edit1.Text+'''';
//form17.Query1.SQL.Clear;
//form17.Query1.SQL.Add(str2);
//form17.Query1.Open;

end;

end.

```

```

unit Unit15;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, DB, DBTables, StdCtrls, Grids, DBGrids, Mask, DBCtrls,
  ExtCtrls,
  ComCtrls;

type
  TForm15 = class(TForm)
    Panel1: TPanel;
    DateTimePicker1: TDateTimePicker;
    Label1: TLabel;
    Button1: TButton;
    DBGrid1: TDBGrid;

    procedure Button1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form15: TForm15;

implementation
  uses unit23, unit25, unit26, unit22;
  {$R *.dfm}

```

```
procedure TForm15.Button1Click(Sender: TObject);
var
  str:string;

begin

  str:=datetostr( DateTimePicker1.Date);
  form22.Table1.Filter:='Date='''+str+''';//and Date='''+edit1.Text+''';
  //code='B'

  form22.Table1.Filtered:=true;

end;

end.
```

```
unit Unit19;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
  Forms,  
  Dialogs, StdCtrls, ExtCtrls, DBCtrls, Mask, DB, DBTables, ExtDlgs,  
  ActnList, DBActns;
```

```
type
```

```
  TForm19 = class(TForm)  
    DBEdit1: TDBEdit;  
    DBEdit2: TDBEdit;  
    DBEdit3: TDBEdit;  
    DBEdit4: TDBEdit;  
    DBEdit5: TDBEdit;  
    DBEdit6: TDBEdit;  
    DBEdit7: TDBEdit;  
    DBRadioGroup1: TDBRadioGroup;  
    DBRadioGroup2: TDBRadioGroup;  
    DBImage1: TDBImage;  
    DBComboBox1: TDBComboBox;  
    Panel1: TPanel;  
    Label1: TLabel;  
    Label2: TLabel;  
    Label3: TLabel;  
    Label4: TLabel;  
    Label5: TLabel;  
    Label6: TLabel;  
    Label7: TLabel;  
    Label8: TLabel;  
    Button1: TButton;  
    Button2: TButton;  
    Button3: TButton;  
    Button4: TButton;  
    Button5: TButton;  
    Button6: TButton;
```



```

DBEdit8: TDBEdit;
Label9: TLabel;
Button7: TButton;
OpenPictureDialog1: TOpenPictureDialog;
Button8: TButton;
Table1: TTable;
DataSource1: TDataSource;
ActionList1: TActionList;
DataSetPost1: TDataSetPost;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure DBImage1Click(Sender: TObject);
procedure Button8Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form19: TForm19;

implementation
    uses clipbrd, unit20;
    {$R *.dfm}

procedure TForm19.Button1Click(Sender: TObject);
begin
    table1.First;
end;

procedure TForm19.Button2Click(Sender: TObject);
begin

```

```
table1.Prior;  
end;
```

```
procedure TForm19.Button3Click(Sender: TObject);  
begin  
table1.Next;  
end;
```

```
procedure TForm19.Button4Click(Sender: TObject);  
begin  
table1.Last;  
end;
```

```
procedure TForm19.Button5Click(Sender: TObject);  
begin  
table1.Edit;  
table1.Insert;  
end;
```

```
procedure TForm19.Button6Click(Sender: TObject);  
begin  
if MessageDlg('Are you sure Want Delete this Record. Exit now?',  
mtConfirmation, [mbYes, mbNo], 0) = mrYes then  
table1.Delete;  
end;
```

```
procedure TForm19.Button7Click(Sender: TObject);
```

```
var  
str:string;  
begin  
if OpenPictureDialog1.Execute then  
begin  
str:=OpenPictureDialog1.FileName;  
dbImage1.Picture.LoadFromFile(str);  
  
end;
```

```
end;
```

```
procedure TForm19.DBImage1Click(Sender: TObject);
```

```
begin
```

```
    DBImage1.Picture.Bitmap.LoadFromFile('c:\2.bmp');
```

```
end;
```

```
procedure TForm19.Button8Click(Sender: TObject);
```

```
begin
```

```
{ if MessageDlg('Are sure you Want Change this Record. Exit now?',
```

```
    mtConfirmation, [mbYes, mbNo], 0) = mrYes then
```

```
    begin
```

```
table1.edit;
```

```
table1.Post;
```

```
end;}
```

```
end;
```

```
end.
```

```
unit Unit20;
```

```
interface
```

```
uses
```

```
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
    Forms,  
    Dialogs, DBCtrls, Grids, DBGrids, StdCtrls, Mask, ExtCtrls, DB,  
    DBTables;
```

```
type
```

```
    TForm20 = class(TForm)  
        Panel1: TPanel;  
        Label1: TLabel;  
        Label2: TLabel;  
        Label3: TLabel;  
        Label4: TLabel;  
        Label5: TLabel;  
        Label6: TLabel;  
        Label7: TLabel;  
        DBEdit1: TDBEdit;  
        DBEdit2: TDBEdit;  
        DBEdit3: TDBEdit;  
        DBEdit4: TDBEdit;  
        DBEdit5: TDBEdit;  
        DBEdit6: TDBEdit;  
        DBEdit7: TDBEdit;  
        DBGrid1: TDBGrid;  
        DBNavigator1: TDBNavigator;  
        Table1: TTable;  
        DataSource1: TDataSource;  
        Table1Code: TStringField;  
        Table1LicenceNO: TStringField;  
        Table1Licencecenter: TStringField;  
        Table1LicenceCat: TStringField;  
        Table1Remarks: TStringField;  
        Table1IssueDate: TDateField;  
        Table1Expdate: TDateField;
```



```

Table2: TTable;
DataSource2: TDataSource;
Table2Code: TStringField;
Table2Name: TStringField;
Table2Surname: TStringField;
Table2Dateofbirth: TDateField;
Table2Placeofbirth: TStringField;
Table2Nameofmother: TStringField;
Table2Gender: TStringField;
Table2Noofpassport: TStringField;
Table2Married: TStringField;
Table2Nationality: TStringField;
Table2Duty: TStringField;
Table2Picture: TGraphicField;
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form20: TForm20;

implementation
    uses unit19;
{$R *.dfm}

end.

```

```

unit Unit21;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, DBCtrls, DB, DBTables, ExtCtrls, StdCtrls, Mask, Grids,
  DBGrids;

type
  TForm21 = class(TForm)
    DBGrid1: TDBGrid;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Panel1: TPanel;
    Table1: TTable;
    DataSource1: TDataSource;
    DBNavigator1: TDBNavigator;
    DataSource2: TDataSource;
    Table2: TTable;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form21: TForm21;

```

implementation

{ \$R \*.dfm }

end.

```

unit Unit23;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, DB, DBTables, StdCtrls, Grids, DBGrids, DBCtrls, ExtCtrls;

type
  TForm23 = class(TForm)
    Panel1: TPanel;
    DBNavigator1: TDBNavigator;
    DBGrid1: TDBGrid;
    ComboBox1: TComboBox;
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form23: TForm23;

implementation
  uses unit15, Unit22;
  {$R *.dfm}

  procedure TForm23.Button1Click(Sender: TObject);
  begin
    if combobox1.Text='Duty' then
    begin

```



```

form22.Table1.Filter:='Duty='''+edit1.text+''';//and
Date='''+edit1.Text+'''; //code='B'
form22.Table1.Filtered:=true;
end
else
begin
    form22.Table1.Filter:='Flight='''+edit1.text+''';//and
    Date='''+edit1.Text+'''; //code='B'
    form22.Table1.Filtered:=true;
end;
    end;
end.

```

```

unit Unit24;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, StdCtrls, DB, DBTables, DBCtrls, Grids, DBGrids, ComCtrls,
  ExtCtrls;

type
  TForm24 = class(TForm)
    Panell: TPanel;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    TabSheet3: TTabSheet;
    DBGrid1: TDBGrid;
    DBGrid2: TDBGrid;
    DBGrid3: TDBGrid;
    DBNavigator1: TDBNavigator;
    DBNavigator2: TDBNavigator;
    DBNavigator3: TDBNavigator;
    Table1: TTable;
    Table2: TTable;
    Table3: TTable;
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    DataSource3: TDataSource;
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Table1Code: TStringField;
    Table1Name: TStringField;
    Table1Surname: TStringField;
    Table1Dateofbirth: TDateField;
    Table1Placeofbirth: TStringField;
    Table1Nameofmother: TStringField;
  end;

```

```

Table1Gender: TStringField;
Table1Noofpassport: TStringField;
Table1Married: TStringField;
Table1Nationality: TStringField;
Table1Duty: TStringField;
Table1Picture: TGraphicField;
Button2: TButton;
Table3Code: TStringField;
Table3LicenceNO: TStringField;
Table3Licencecenter: TStringField;
Table3LicenceCat: TStringField;
Table3IssueDate: TDateField;
Table3Expdate: TDateField;
Table3Remarks: TStringField;
Table2Code: TStringField;
Table2PhoneNO1: TStringField;
Table2PhoneNO2: TStringField;
Table2Address1: TStringField;
Table2Address2: TStringField;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form24: TForm24;

implementation

{$R *.dfm}

procedure TForm24.Button1Click(Sender: TObject);
begin

```

```
Table1.Filter:='Code='''+edit1.text+''';//and Date='''+edit1.Text+''';  
//code='B'  
Table1.Filtered:=true;
```

```
Table2.Filter:='Code='''+edit1.text+''';//and Date='''+edit1.Text+''';  
//code='B'  
Table2.Filtered:=true;
```

```
Table2.Filter:='Code='''+edit1.text+''';//and Date='''+edit1.Text+''';  
//code='B'  
Table2.Filtered:=true;
```

```
end;
```

```
procedure TForm24.Button2Click(Sender: TObject);  
begin  
table1.Edit;  
table1.UpdateRecord;
```

```
end;
```

```
end.
```



```

unit Unit25;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, DB, StdCtrls, DBTables, ExtCtrls, QuickRpt, QRCtrl;

type
  TForm25 = class(TForm)
    QuickRep1: TQuickRep;
    PageHeaderBand1: TQRBand;
    TitleBand1: TQRBand;
    ColumnHeaderBand1: TQRBand;
    DetailBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRLabel2: TQRLabel;
    QRLabel3: TQRLabel;
    QRDBText1: TQRDBText;
    QRLabel4: TQRLabel;
    QRLabel5: TQRLabel;
    QRLabel6: TQRLabel;
    QRLabel7: TQRLabel;
    QRLabel8: TQRLabel;
    QRLabel9: TQRLabel;
    QRLabel10: TQRLabel;
    QRDBText2: TQRDBText;
    QRDBText3: TQRDBText;
    QRDBText4: TQRDBText;
    QRDBText5: TQRDBText;
    QRDBText6: TQRDBText;

    QRDBText7: TQRDBText;
    QRDBText8: TQRDBText;
    QRLabel11: TQRLabel;
    QRSysData1: TQRSysData;
  end;

```

```
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form25: TForm25;

implementation
  uses unit17,unit15,unit26;
  {$R *.dfm}

end.
```