NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

HOTEL DATABASE DESIGN

Graduation Project
COM-400

Student: Razib Ahmed (992051)

Supervisor: Assist. Prof. Dr. Adil Amirjanov

Nicosia-2006

ACKNOWLEDGMENT

First of all I would like to thank Assist. Prof. Dr. Adil Amirjanov for his endless and untiring support and help and his persistence, in the course of the preparation of this project.

Under his guidance, I have overcome many difficulties that I faced during the various stages of the preparation of this project.

I would like to thank all of my friends who helped me to do the project according to the necessity especially Mohammad Abdur Rob.

Finally, I would like to thank my family, especially my parents. Their love and guidance made me to come at this end of my study. Their never-ending belief in me and their encouragement has been a crucial and a very strong pillar that has held me together.

They have made countless sacrifices for my betterment. I can't repay them, but I do hope that their endless effort will bear good fortune for me that may lead them, me and all who surround me to a better future.

ABSTRACT

In this project development of software on 'Hotel' has been considered. This software complied with Microsoft Access Program which is a database program included Microsoft Office Pack.

Nowadays, computers are used almost every area of business and life, with computers and software they increased the speed of our calculation, transaction, necessary information saving ...etc.

This application is designed for 'Hotel' which cover all the needs for the hotel business as an administrative employee like 'Manager'. In this program it's able to hold customer records, employee records, accounts, expenses, needs, transaction and we can get the report of all individual sides very easily.

With this program maintenance of a hotel will be very easy, faster and reliable.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

A database is a means of collecting and organizing information. You can create simple ones that contain a list of your business contacts, for example, or you can build a full-featured data management system that you can use to manage a business. Microsoft Access gives us the tools to build just about any kind of database we need.

Access is a popular development platform in large measure because it is the part of the Microsoft Office suite. Many clients want their Access systems to interoperate with the rest of Office, and they want systems that are transparent and easy to maintain without developer assistance.

Once we have created a database and added information to it, we can use Access to create forms or reports. With these we can retrieve the information we've put into the database. With the program's built-in samples, templates, and wizards, it's easy for a beginner to get started. Access 2000 is straightforward enough to allow a beginner to create quick and simple, but very useful databases. As our skills grow, we can add more features that will expand our database's usefulness to our business.

In the project development of database system was considered for **'Hotel'** with Microsoft Access programming.

In **Chapter one:** we discussed about database, data model, type of databases, relationships, primary key, foreign key and compound key in details.

In **Chapter two:** general Sql structure and its most used keywords presented.

In **Chapter three:** introductory remarks on Microsoft Access programming are given. Creating and modifying of Tables, Queries, Reports, Forms are explained. How to use wizard is illustrated with figures.

In **Chapter four:** general structure of our application is given. Implementation of the program is given step by step with explanation.

CHAPTER ONE: INTRODUCTION TO DATABASE MANAGEMENT SYSTEMS

1.1 Overview

A database is a collection of information stored in a computer in a systematic way, such that a computer program can consult it to answer questions. The software used to manage and query a database is known as a database management system (DBMS). The properties of database systems are studied in information science.

At the core of the concept of a database is the idea of a collection of facts, or pieces of knowledge. Facts may be structured in a number of ways, known as data models. For instance, one model is to associate each fact with a record representing an entity (such as a person), and to arrange these entities into trees or hierarchies, which is called the hierarchical data model. Another model is to arrange facts into sets of values which satisfy logical predicates, which is called the relational data model.

Database management systems are ranged from extremely simple to highly complex. Working from a blank page, creating a database application from scratch is a difficult task. In fact, before we can do any customizing or programming, we have to get most of our database in place and working. Fortunately, Microsoft Access includes several wizards that can help us to get started quickly. Differences among DBMSes includes whether they are capable of ensuring the integrity of the data; whether they may be used by many users at once; and what sorts of conclusions they can be programmed to compute from a set of data.

The terms database and database management system are sometimes interchanged by students. In the professional area, a database is always the collection of facts, not the software program.

The first database management systems were developed in the year 1960. A pioneer in the field was Charles Bachman. Two key data models arose at this time: the network model followed by the hierarchical model. These were later usurped by the relational model, which was contemporary with so-called flat model designed for very small tasks. Another contemporary of the relational model is the object-oriented database (OODB). While the relational model is based on the set theory, one proposed modification suggests fuzzy set theory as an alternative.

1.2 Database Models

Various techniques are used to model data structures. Certain models are more easily implemented by some types of database management systems than others. For any one logical model various physical implementation may be possible. An example of this is the relational model: in larger systems the physical implementation often has indexes which point to the data; this is similar to some aspects of common implementations of the network model. But in small relational database the data is often stored in a set of files, one per table, in a flat, un-indexed structure. There is some confusion below and elsewhere in this article as to logical data model vs its physical implementation.

1.2.1 Flat Model

The flat (or table) model consists of a single, two dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another. For instance, columns for name and password might be used as a part of a system security database. Each row would have the specific password associated with a specific user. Columns of the table often have a type associated with them, defining them as character data, date or time information, integers, or floating point numbers. This model is the basis of the spreadsheet.

1.2.2 Network Model

The network model allows multiple datasets to be used together through the use of pointers (or references). Some columns contain pointers to different tables instead of data. Thus, the tables are related by references, which can be viewed as a network structure. A particular subset of the network model, the hierarchical model, limits the relationships to a tree structure, instead of the more general directed graph structure implied by the full network model.

1.2.3 Relational Model

The relational data model was introduced in an academic paper by E.F. Codd in 1970 as a way to make database management systems more independent of any particular application. It is a mathematical model defined in terms of predicate logic and set theory.
Although the basic idea of a relational database has been very popular, relatively few people understand the mathematical definition and only a few obscure DBMSs implement it completely and without extension. Oracle, for example, can be used in a purely relational way, but it also allow tables to be defined that allow duplicate rows an extension (or violation) of the relational model. In common English usage, a DBMS is called relational if it supports relational operational operations, regardless of whether it enforces strict adherence to the relational model. The following is an informal, not-technical explanation of how "relational" database management systems commonly work.

A relational database contains multiple tables, each similar to the one in the "flat" database model. However, unlike network databases, the tables are not linked by pointers. Instead, keys are used to match up rows of data in different tables. A key is just one or more columns in one table that correspond to columns in other tables. Any column can be a key, or multiple columns can be grouped together into a single key. Unlike pointers, it's not necessary to define all the keys in advance; a column can be used as a key even if it wasn't originally intended to be one.

A key that can be used to uniquely identify a row in a table is called a unique key. Typically one of the unique keys is the preferred way to refer to row; this is defined as the table's primary key.

When a key consists of data that has an external, real-world meaning (such as a person's name, a book's ISBN, or a car's serial number), it's called a "natural" key. If no nature key is suitable, an arbitrary key can be assigned (such as by given employees ID numbers). In practice, most databases have both generated and natural keys, because generated keys can be used internally to create links between rows that can't break, while natural keys can be used, less reliably, for searches and for integration with other databases. (For example, records in two independently developed databases could be matched up by social security number, except when the social security numbers are incorrect, missing, or have changed).

1.3 Why we use a Relational Database Design

Maintaining a simple, so-called flat database consisting of a single table doesn't require much knowledge of database theory. On the other hand, most database worth maintaining are quite a bit more complicated than that. Real life databases often have hundreds of thousands or even millions of records, with data that are very intricately related. This is where using a full-fledged relational database program becomes essential. Consider, for example, the Library of Congress, which has over 16 million books in its collection. For reasons that will become apparent soon, a single table simply will not do for this database.

1.3.1 Relationships Between Tables

When you create tables for an application, you should also consider the relationships between them. These relationships give a relational database much of its power. There are three types of relationships between tables: one-to-one, one-to-many and many-to-many relationships.

1.3.2 One-To-One Relationships

In a one-to-one relationship, each record in one table corresponds to a single record in a second table. This relationship is not very common, but it can offer several benefits. First, you can put the fields from both tables into a single, combined table. One reason for using two tables is that each field is a property of a separate entity, such as owner operators and their tracks. Each operator can operate just one truck at a time, but the fields for the operator and truck tables refer to different entities.

A one-to-one relationship can also reduce the time needed to open a large table by placing some of the table's columns in a second, separate table. This approach makes particular sense when a table has some fields that are used infrequently. Finally, a one-to-one relationship can support in a table requires security, placing them in a separate table lets your application restrict to certain fields. Your application can link the restricted table back to the main table via a one-to-one relationship so that people with proper permissions can edit, delete, and add new records to these fields.

1.3.3 One-To-Many Relationships

A one-to-many relationship, in which a row from one table corresponds to one or more rows from a second table, is more common. This kind of relationship can form the basis for a **Many-To-Many** relationship as well.

Redundancy

The main problems associated with using a single table to maintain a database system from the issue of unnecessary repetition of data, that is, redundancy. Some repetition of data is always necessary, but the idea is to remove as much unnecessary repetition as possible. The redundancy in the library flat table (Table 1.1) is obvious. For instance, the name and phone number of Big House publishers is repeated in the table. In an effort to remove as much redundancy as possible from a database, a database designer must split the data into multiple tables. Here is one possibility for the library_flat example, which splits the original database into four separate tables.

- A books table, shown in table 1.2, in which each book has its own record
- An authors table, shown in table 1.3, in which each author has his or her own record

- A publishers table, shown in table 1.4, in which each publisher has its own record
- Book/author table, shown in table 1.5, the purpose of which we will explain a bit later

To get a feel for the reduction in duplicate data achieved by the four-table approach, imagine (as it reasonable) that the database also includes the addresses of each publisher. Then table 1.1 would need a new column containing many addresses many of which are duplicates. On the other hand, the four-table database needs only one new column in the publishers table, adding a total of three distinct addresses. To drive the difference home, consider the 16 million-book database of the Library of Congress. Suppose the database contains books from 10,000 different publishers. A publisher's address column in a flat database design would contain 16 million addresses, whereas a multi-table approach would require only 10,000 addresses. Now, if the average address is 50 characters long, then the multi-table approach would save. (16,000,000-10,000)*50= 799 million characters.

Assuming that each character takes 2 bytes (in the Unicode that is used internally by Microsoft Access), the single-table approach wastes about 1.6 gigabytes of space, just for the address field. Indeed, the issue of redundancy alone is quite enough to convince a database designer to avoid the flat database approach. However, there are several other problems with flat databases, which we now discuss.

1.4 Relational Operations

You request data from a relational database by sending it a query that's written in a special language, usually a dialect of SQL. Although SQL was originally intended for end-users, it's much more common for SQL queries to be embedded into software that provides an easier user interface.

In response to a query, the database returns a results set, which is just a list of rows containing the answers. The simplest query is just to return all the rows from a table, but more often, the rows are filtered in some way to return just the answer wanted.

Often, data from multiple tables gets combined into one, by doing a "join". Conceptually, this is done by taking all possible combinations of rows (the "cross-product"), and then filtering out everything except the answer. In practice, relational database management systems rewrite ("optimize") queries to perform faster, using a variety of techniques. The flexibility of relational databases allows programmers to write queries that were not anticipated by the database designer. As a result, relational databases can be used by multiple applications in ways the original designers didn't foresee, which is especially important for databases that might be used for decades. This has made the idea and implementation of relational databases very popular with businesses.

1.4.1 Implementations and indexing

All of these kinds of database can take advantage of indexing to increase their speed. The most common kind of index is a sorted list of the contents of some particular table column, with pointers to the row associated with the value. An index allows a set of table rows matching some criterion to be located quickly. Various methods of indexing are commonly used; b-tree, hashes, and linked lists are all common indexing techniques.

Relational DBMSs have the advantage that indices can be created or dropped without changing existing applications, because applications don't use the indices directly. Instead, the database software decides on behalf of the application which indices to use. The database chooses between many different strategies based on which one it estimates will run the fastest.

Relational DBMSs utilize many different algorithms to compute the result of an SQL statement. The RDBMS will produce a plan of how to execute the query, which is generated by analyzing the run times of the different algorithms and selecting the quickest. Some of the key algorithms that deal with joins are Nested Loops Join, Sort-Merge Join and Hash Join.

1.5 Application Of Database

Databases are used in many applications, spanning virtually the entire range of computer software. Databases are the preferred method of storage for large multi-user applications, where coordination between many users is needed. Even individual users find them convenient, though, and many electronic mail programs and personal organizers are based on standard database technology.

A database management system (DBMS) is a computer program (or more typically, a suite of them) designed to manage a database; a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMS use include accounting, human resources and customer support systems. Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

DBMS's are found at the heart of most database applications. Sometimes DBMSs are built around a private multitasking kernel with built-in networking support although nowadays these functions are left to the operating system.

1.6 Data Modeling

In information system design, data modeling is the analysis and design of the information in the system, concentrating on the logical entities and the logical dependencies between these entities. Data modeling is an abstraction activity in that the details of the values of individual data observations are ignored in favor of the structure, relationships, names and formats of the data of interest, although a list of valid values is frequently recorded. It is by the data model that definitions of what the data means is related to the data structures.

While a common term for this activity is "Data Analysis" the activity actually has more in common with the ideas and methods of **synthesis** (putting things together), than it does in the original meaning of the term **analysis** (taking things apart). This is because the activity strives to bring the data structures of interest together in a cohesive, inseparable, whole by eliminating unnecessary data redundancies and relating data structures by relationships. In the early phases of a software development project, emphasis will be on the design of a conceptual data model. This can be detailed into a logical data model sometimes called functional data model. In later stages, this model may be translated into physical data model.

1.6.1 Database Normalization

Database normalization is a series of steps followed to obtain a database design that allows for consistent storage and efficient access of data in a relational database. These steps reduce data redundancy and the risk of data becoming inconsistent.

However, many relational DBMSs lack sufficient separation between the logical database design and the physical implementation of the data store, such that queries against a fully normalized database often perform poorly. In this case de-normalizations is sometimes used to improve performance, at the cost of reduced consistency.

1.6.2 Primary Key

In database design, a **primary key** is a value that can be used to identify a particular row in a table. Attributes are associated with it. Examples are names in a telephone book (to look up telephone numbers), words in a dictionary (to look up definitions) and Dewey Decimal Numbers (to look up books in a library).

In the relational model of data, a **primary key** is a candidate key chosen as the main method of uniquely identifying a relation. Practical telephone books, dictionaries and libraries can not use names, words or Dewey Decimal System Numbers as candidate keys because they do not uniquely identify telephone numbers, word definitions or books. In some design situations it is impossible to find a natural key that uniquely identifies a relation. A **surrogate key** can be used as the **primary key**. In other situations there may be more than one candidate key for a relation, and no candidate key is obviously preferred. A surrogate key may be used as the primary key to avoid giving one candidate key artificial primacy over the others. In addition to the requirement that the primary key be a candidate key, there are several other factors which may make a particular choice of key better than others for a given relation.

The **primary key** should generally be short to minimize the amount of data that needs to be stored by other relations that reference it. A compound key is usually not appropriate. (However, this is a design consideration, and some database management systems may be better than others in this regard.)

The **primary key** should be immutable, meaning its value should not be changed during the course of normal operations of the database. (Recall that a **primary key** is the means of uniquely identifying a tuple, and that identity by definition, never changes.) This avoids the problem of dangling references or orphan records created by other relations referring to a tuple whose primary key has changed. If the **primary key** is immutable, this can never happen.

1.6.3 Foreign Key

A **foreign key (FK)** is a field in a database record under one primary key that points to a key field of another database record in another table where the **foreign key** of one table refers to the **primary key** of the other table. This way references can be made to link information together and it is an essential part of database normalization.

For example, a person sending an e-mail needs not to include the entire text of a book in the e-mail. Instead, they can include the ISBN of the book, and interested persons can then use the number to get information about the book, or even the book itself. The ISBN is the **primary key** of the book, and it is used as a **foreign key** in the e-mail.

Note that using a **foreign key** often assumes its existence as a **primary key** somewhere else. Improper **foreign key/primary key** relationships are the source of many database problems.

1.6.4 Compound Key

In database design, a **compound key** (also called a **composite key**) is a key that consists on 2 or more attributes.

No restriction is applied to the attribute regarding their (initial) ownership within the data model. This means that any one, none or all, of the multiple attributes within the **compound key** can be **foreign keys**. Indeed, a foreign key may, itself, be a compound key.

Compound keys almost always originate from attributive or associative entities (tables) within the model, but this is not an absolute value.

CHAPTER TWO: STRUCTURED QUERY LANGUAGE (SQL)

Structured Query Language (SQL) is the most popular computer language used to create, modify and retrieve data from relational management systems. The language has evolved beyond its original purpose to support object-relational database management systems.

A seminal paper, "A Relational Model of data for Large Shared Data Banks, by Dr. Edgar F. Codd, was published in June, 1970 in the Association for Computing Machinery (ACM) journal, Communications of the acm." Codd's model became widely accepted as the definitive model for the relational database management system (RDBMS).

During the 1970s, a group at IBM's San Jose research center developed a database system "System R" based upon Codd's model. **Structured English Query Language ("SEQUEL")** was designed to manipulate and retrieve data stored in System R. The acronym sequel was later condensed to SQL due to a trademark dispute (the word "sequel" was held as a trade mark by the Hawker-Siddeley aircraft company of the UK).

In 1979, Relational Software, Inc. (now Oracle Corporation) introduced the first commercially available implementation of SQL and, soon many vendors developed dialects of it.

SQL was adopted as a standard by the ANSI (American National Standard Institute) in 1986 and IOS (International Organization for Standardization) in1987. ANSI has declared that the official pronunciation for SQL is "es queue el", although many English-speaking database professionals still pronounce it as SEQUEL.

2.1 Description of SQL

SQL allows the specification of queries in a high level, declarative manner. For example, to select rows from a database, the user need only specify the criteria that want to search by; the details of performing the search operation efficiently is left up to the database system, and is invisible to the user.

Compared to general purpose programming languages, this structure allows the user/programmer to be less familiar with the information contained in the data. This blurs the line between user and programmer, appealing to individuals who fall more into the ‘business’ or ‘research’ area and less in the ‘information technology’ area. The original vision for SQL was to allow non –technical users to write their own database queries. While this has been realized to some extent, the complexity of querying an advanced database system using SQL can still require a significant learning curve.

SQL contrasts with the more powerful database-oriented fourth-generation programming languages such as focus or sas, however, in its relative functional simplicity and simpler command set. This greatly reduces the degree of difficulty involved in maintaining the worst SQL source code, but it also makes programming such questions as ‘Who had the top ten scores?’ more difficult, leading to the development of procedural extensions, discussed above. However, it also makes it possible for SQL source code to be produced (and optimized) by software, leading to the development of a number of natural language database query languages, as well as ‘drag and drop’ database programming packages with ‘object oriented’ interfaces. Often these allow the resultant SQL source code to be examined, for educational purposes, further enhancement, or to be used in a different environment.

2.2 SQL Keywords

SQL keywords fall into several groups, like:

2.2.1 Data Retrieval

The most frequently used operation in transactional databases are the data retrieval operation.

- SELECT is used to retrieve zero or more rows from one or more tables in a database. In most applications, SELECT is the most commonly used DML command. In specifying a select query, the user specifies a description of the desired result set, but they do not specify what physical operations must be executed to produce that result set. Translating the query into an optimal query plan is to leave the database system, more specifically to the query optimizer.

- Commonly available keywords related to SELECT includes:

  - FROM is used to indicate which tables the data is to be taken from, as well as how the tables join to each other.
  - WHERE is used to identify which rows to be retrieved, or applied to GROUP BY.
  - GROUP BY is used to combine rows with related values into elements of a smaller set of rows.
  - ORDER BY is used to identify which columns are used to sort the resulting data.

2.2.2 Data Manipulation

First there are the standard Data Manipulation Language (DML) elements. DML is the subset of the language used to add, update and delete data.

- INSERT is used to add the zero or more rows (formally tuples) to an existing table.
- UPDATE is used to modify the values of a set of existing table rows.
- DELETE removes zero or more existing rows from a table.

2.2.3 Data Transaction

Transaction, if available, can be used to wrap around the DML operations. BEGIN WORK (or START TRANSACTION, depending on SQL dialect) can be used to mark the start of a database transaction, which either completes completely or not at all
COMMIT causes all data changes in a transaction to be made permanent.

ROLLBACK causes all data changes since the last COMMIT or ROLLBACK to be discarded, so that the state of the data is "Rolled Back" to the way it was prior to those changes being requested.

COMMIT and ROLLBACK interact with areas such as transaction control and locking. Strictly, both terminate any open transaction and release any locks held on data. In the absence of a BEGIN WORK or similar statement, the semantics of SQL are implementation-dependent.

2.2.4 Data Definition

The second group of keywords is the Data Definition Language (DDL). DDL allows the user to define new tables and associated elements. Most commercial SQL databases have proprietary extensions in their DDL, which allow control over nonstandard features of the database system.

The most basic items of DDL are the CREATE and DROP commands. CREATE causes an object (a table, for example) to be created within the database. DROP causes an existing object within the database to be deleted, usually irretrievably. Some database systems also have an ALTER command, which permits the user to modify an existing object in various ways, for example, adding a column to an existing table.

CHAPTER THREE: MICROSOFT ACCESS DATABASE SYSTEM

3.1 Introductory Microsoft Access

- Starting Microsoft Access
- Opening Access
- Click Start, select All Programs and click on Microsoft Access
- Creating a Database
- Click File, New or click the new icon on the standard toolbar
- Select Blank Database from the Task Pane menu
- Type a name for the Database in the File Name window
- Click Create
- Closing a database click File, Close
- Opening a database click File, Open or click the open icon on the standard toolbar
- Browse, where to save the database
- Click the name of the database
- Click Open

3.2 Working with Tables

A table is a collection of data about a specific topic, such as products, employees, reservation or companies.

Creating a table:

- From the main database window, click **Tables** under **Objects** on the left menu
- Create a table without any assistance, like:
  - Double click on **Create table in design view**
  - In the **Field Name** column type the name of data field (i.e. MailingListID)
  - In the **Data Type** column select the type of data to be entered in the field (i.e. Text, Number, OLE Objects etc)
  - To save the table, click **File**, then click **Save** or click the save icon on the Standard toolbar
  - Type the name of the **Table**

- Click OK
- If we do not want any **Primary Key**, click no
- We can enter the data in the table straight or by using the form

Creating a table using the wizard:

- Double click on **Create table by using wizard**
- Select the type of table (**Business** or **Personal**)
- Choose a table from the **Sample Tables** list
- Select a data field to include in the table
- Click the **single arrow**
- Repeat these steps or if we want to get all the fields to include we have to click on the **double arrow**
- Click **Next** if we want to make any field as the Primary Key field
- Then click **Next** if we want to make any **form** for the table to enter the data straight in the table
- Click **Finish**
- Enter data into table straight or by using the **form**

3.3 Creating Tables Manually

To create a table manually, we double click on the 'Create Table In Design View' icon in the database window to open a blank table window in the design view. From here, we can add any fields, a Primary or a Composite Key or an index.

To add a field, type the field's name in a blank field name column in the Design view window. Field names follow normal Visual Basic for Applications (VBA) naming conventions. They can be up to 64 characters long, and the characters can be letters, numbers, spaces, and special characters except the period, the exclamation mark, square brackets and the grave accent character ('). Also we can not start a field name with a space or a control character (ASCII values 0 through 31). While we can include internal spaces in field names, they must be bracketed in expressions and queries.

3.3.1 Data Types

In the data type column, we can specify a data type for the field. A drop down list box offers 10 options: **Text, Memo, Number, Date/Time, Currency, AutoNumber, Yes/No, OLE Object, Hyperlink and Lookup Wizard**. (Data types are commonly used to identify the information a field contains. A **Text** field contains a Text data type; an **AutoNumber** field contains an AutoNumber data type and so on.) We can use the options within many of these data types to further refine our data type specifications.

3.3.2 AutoNumber Fields

AutoNumber field types frequently serve as the Primary Key for a table. Access automatically assigns a new value to the field when we add a record to the table. This field is not manually updateable, so its values are ideal for uniquely marking a row within a table

Access automatically sets the value of AutoNumber field types. To cause an AutoNumber field to increment sequentially, select Increment (the default value) from the New Values drop-down list box on the General page at the bottom left of the table window. To indicate that an AutoNumber field should have a randomly assigned value, select Random from the New Values drop down list box.

We can use the General and Lookup pages to select other properties that affect the field, such as whether the user must enter a value into the field or whether the field has a default value. Field properties set as the table level propagate through to forms and reports. Maintaining data properties at the table level also means that properties are changed in a single place rather than within each form and report that uses a field.

Access 2000 is the first version of Access to enable programmatic control over the initial value and step size of AutoNumber field type. We can use the Alter Table and Alter Column keywords in Jet SQL to update the next start. The start and step properties of this data type let us programmatically modify the next AutoNumber value and the step size for subsequent values.

3.3.3 Text Fields

We use Text Fields to hold string entries that contain up-to 255 characters. The Text data type can store items such as contact information and numerical data values that do not require computation (for example, telephone numbers, addresses). We can also use Text Fields in a table to persist computed string values. We can index Primary Keys for fast sorts and retrieval based on last name or another Text Field type.

3.3.4 Number Fields

Number Fields are different from Text Fields because they can assume a variety of subtypes ranging from a single byte (Byte subtype) to 16 bytes (Replication ID subtype). The other data subtypes between these extremes include Integers, Long Integer, Single, Double and Decimal. With the exception of the Byte and Replication ID subtypes. The Byte subtype is similar to the Boolean variable data type. Both types can store Boolean values, but the Byte subtype requires just 1 byte of storage while the Boolean data type 2 bytes. The Replication ID data type is not available as a variable data type. Its primary use is in replication, but it serves as a unique identifier. Its length and method of creation make it a more secure way to ensure uniqueness than an AutoNumber field.

The decimal subtype facilitates the elimination of rounding errors while still accommodating large numbers using Precision and Scale properties. These properties control the number of digits on either side of the decimal point. Precision, which represents the total number of digits that can be stored in the field, can range from 1 through 28. Scale, which indicates the number of digits to the right of the decimal that can be stored in the field, can range from 0 through the value in the Precision property. Because of the Scale property, the Decimal data subtype can store more digits after the decimal point without rounding errors than other Number data subtypes can.

The CurrencyBalance field uses the Currency data type; CurrencyFloat uses the Number data type with the Double subtype and CurrencyBalanceDec field uses the Number data type with the Decimal subtype. The CurrencyBalanceDec field has a Scale property setting of 6, which indicates that the field can store six digits to the right of the decimal point. This is more digits than the Currency data type can precisely present its limit is four digits after the decimal. The Double data subtype can represent a number with four, five or six places after the decimal, but it does not perform this task with integer precision. The first row in persons displays the value 1.0001 in Currency, Double and Decimal data formats. The second row expresses 1.00001 in the same three formats. Notice that in the Datasheet view the Currency format initially shows 1.00001 as 1.0000 since it is limited to four places after the decimal. The Double and Decimal representations appear identical.

3.3.5 Memo, OLE object, Date/Time and Yes/no Fields

Other data types includes the Memo data type, which holds very large text data strings that can exceed the 255 characters limit of the text data type. A single Memo data type can grow to 64 KB. We can access and write back its contents in 64 KB blocks using the GetChunk and AppendChunk methods. Jet 4 supports indexing the first 255 characters of a Memo field. This is particularly useful for Hyperlink data types that depend on the Memo data type.

OLE Object is another large data type. It works with objects in their binary format, such as a Microsoft Excel workbook or a Microsoft Word document.

Date/Time data types can represent either Dates or Times. Date values are stored to the left of the decimal point, Time values are stored to the right of the decimal point.
The Yes/No data type is the smallest data area. It always in one of two states, either Yes/No, True/False or On/Off. It occupies a single byte of storage.

3.3.6 Primary Key Field

A Primary Key field is a field or combination of fields that uniquely identify each record in a table. Primary Key features:

- In a table two records can not have the same value in the Primary Key field. Records are automatically stored based on the Primary Key.

Primary Keys performs the following functions:

- Prevent duplicate values
- Maintains the record order
- Creates a Primary index. Indexes are used to improve the speed of queries, reports, and locating records
- Facilities relationships to other normalized data in the database. The Primary table to be joined must have a Primary Key field.

NOTE: Memo, Yes/No, OLE Objects and Hyperlink fields can not be Primary Key fields.

Entering data in the Table:

- Double click on the table name
- Click in the field you wish to enter data
- Press Tab to move to the next field or press Enter to move to the next record sorting data
- Click in the data field we wish to sort by
- Click a sort icon on the Standard Toolbar
- Or click Records, Sort
- Select the sort order (Ascending or Descending)

Deleting data from a Table:

- Click in the record we wish to delete
- Click Edit, Delete Record

Saving data in a Table:

- Click File, Save or click the save icon on the Standard Toolbar

Closing a Table:

- Click File, Close. Or click on the cross on the upper right corner of the window.

3.3.7 Forms-Creating and Working with Forms

While the Access Project user interface delivers extraordinary functionality with remote data sources, we can automate and simplify processes by developing custom programmatic solutions.

Opening a form

When we open a form with the Access Project interface, the form populates a local copy of the remote data in the client workstation. This local copy is a snapshot, at a point in time, of the remote data for the form. When we open the form, we must create the local copy of the remote data. One advantage to opening a form in the program is that we can dynamically assign values to the local cache of the remote data. Our application can do this because the record set that we assign to the form with visual basic for applications overrides the record source setting on the form's property sheet.

The procedure below constructs a record set for a form before opening it. It starts by setting a reference to a new record set instance: it assigns adUseClient to the record set's CursorLocation property to establish the location of the form's data. Next it opens the record set with a SQL statement that extracts data from the remote source into the local copy. A commented line shows a SQL statement that can override the form's default record source. After making the local copy of the remote data, the procedure opens the form and assigns the local copy to the form's record set property. This new property processes the functionality of the RecordsetClone property; in addition, changes to the Recordset appear on the form automatically; the RecordsetClone property provides a read-only copy of a form's Recordset. Like the RecordsetClone property, a form's Recordset property is available only programmatically.

3.3.8 Switchboard Forms

The 'Hotel' database we have created is more than an ordinary Access database: with its Switchboard form and the other forms and reports it contains, the 'Hotel' database is an application. This term refers to the fact that the database has its own custom-built user interface, designed especially to help users of the database navigate through its forms and reports and gets work done with the database. Database applications range from those as simple as the 'Hotel' database, which has a very specific job, to complete business solutions that contain many objects and large amounts of Visual Basic code.

IMPORTANT: The Switchboard Form that the wizard creates is a special form that we should not try to change directly. It contains Visual Basic code that, along with the Switchboard Items table, displays the text we see on the form and makes the buttons work. Although we can not easily modify the Switchboard Form work, we can add or remove buttons.

3.3.8.1 Subforms

A Subform, one of the most popular ways of displaying data in Access, is a form embedded within a main form. The main form holds general information about an object (such as an accounts or a reservation). One or more hierarchically related details (such as a expenses record or room information) appear in one or more Subforms on the main from. At least one common field must tie the record source of the main form and each Subform together. The common field enables the Subform to show only records that match the current record in the main form. When the user moves to a new record on the main form, the Subform displays a new set of records that tie uniquely to the new record in the main form.

To create a Subform, open the main form in design view, make sure that the Control Wizard button is selected on the Toolbox, and then drag a table, query or form from the database window and drop it on the main form. The Subform appears as a control on the main form. To synchronize the main form and the Subform, we must designate at least one common field. Select the Subform container and set its Link Child and Link Master properties to the common field.

A main form can have multiple Subforms. The only requirements that the source for each Subform share at least one common field with the record source for the main form. If we define relationships between tables and queries in the Relationships window or by using the properties of a Subdatasheet, we can create a main form with an embedded Subform as easily as we create a simple bound form. In the Database window, select the table or query on which the main form will be based, and then click the New Object; AutoForm button. The AutoForm wizard will build a main form with an embedded Subform. The Subform uses the information in the Relationships windows or the Subdatasheet. We can manually drag other tables, queries or forms to the main form in Design view to create additional Subform.

Creating a Form

- From the main database window, click **Forms** under **Objects** on the left menu
- Click **Create form by using wizard**
- In the **Tables/Queries** drop down list, select the table or query we wish to create the form from
- In the **Available Fields** window, select a data field to include in the form and click the **Single arrow** to select one data field in the Selected Fields
- Or click the **Double arrow** to include all data fields in the form's field
- Click **Next**
- Select a layout for the form, the click **Next**
- Select a style for the form, then click **Next**
- Type a name for the **Form**
- Click **Finish**

Entering data in a form

- Open a Form by double clicking on a form name
- Click in a data field and enter the data
- Click tab to move to the next data field
- To move between records, use the arrows on the record menu at the bottom of the form
- To insert a new record, click the arrow with the star sign (*) beside it

Changing the design of a form

- Click **View**, **Design View** or click the **View** icon on the **Standard Toolbar**
- To change the style, click **Format**, **AutoFormat**
- Select the new format, then click **OK**
- To move form objects, position the mouse over the object. When the **hand** sign appears, click and hold the mouse button down then move the object to its new location
- To change the color of an object, right click on the object and select a new fill or font color
- To add any new **button** to the form, click the toolbox, check the toolbox's **wizard** button is selected. Then click on the button sign in the toolbox, drag it to the **footer** of the form and then in the designated location click the mouse. The wizard setting will start and will ask to select the button type. After choosing the type we can give a name to the button or can set a picture icon on it
- Then click **Finish**

Saving a Form

- Click **File**, **Save** or click the **save icon** on the Standard Toolbar

Closing a Form

- Click **File**, **Close**

Working with Queries

We use queries to view, change and analyze data in different ways. We can also use them as a source of records for forms and reports. There are five types of queries: **Select**, **Parameter**, **Crosstab**, **Action** and **SQL**.

3.3.8.2 Parameter Queries

Parameter queries are a special type of query that can return rows or perform actions. At run time, a parameter query can prompt the user for input that controls how it performs. We can prompt for one or more inputs by using different data type specifications. We tell the parameter query what to do by inputting values to its prompts or by setting its parameters with visual basic for applications code before executing the query to control the return set or action that it performs. This allows the designation of a customer ID value at run time to determine the customer about which a select query returns information.

As an alternative to a parameter query, our application can reference a SQL string for a select or action query with string variables. Before executing the SQL statement in an ActiveX Data Objects (ADO) command, assign the string variables specific values. This can provide more flexible results than a parameter query since we can actually alter whole clauses in the SQL statement for a query. For certain cases, parameter queries offset these benefits with data typing and built-in prompts for values. In addition, parameter queries eliminate the need to refine string concatenation statement as we refine our query's SQL statement.

3.4 Creating a Query

- From the main database window, click **Queries** under **Objects** on the left menu
- Double click on the **Create query by using wizard** option
- In the **Tables/Queries** drop-down list, select the table or query we wish to create for query

- In the **Available Fields** window, select a data field. To select the field click the **single arrow**
- Or to select **all the fields** we need to click the **double arrow**, which will include all the fields in the query. We can chose fields from more than one table or query.
- Click **Next**
- Type a name for the query, then click **Finish**

3.4.1 Restricting Query Data

- Click **View**, **Design** or click the view icon on the Standard Toolbar
- Click in the **criteria box** of the field we wish to restrict
- Enter the value for the **criteria**
- Click **View**, **Datasheet View** or click the view icon on the Standard Toolbar to view the query

3.4.2 Saving a Query

- Click **File**, **Save** or click on the save icon on the Standard Toolbar

3.4.3 Closing a Query

- Click **File**, **Close**

3.5 Working with Reports

We use reports to view, organize and summarize data in a printable format

3.5.1 Creating a Report

- From the main database window, click **Reports** under **Objects** on the left menu
- Double click on the **Create report by using wizard** option
- In the **Tables/Queries** drop-down list, select the table or query we wish to create the report from
- In the **Available Fields** window, select a data field to include in the report and click the **single arrow**
- Or click the **Double arrow** to include **all the data fields** to the report

- Click **Next**
- Click **Next**
- In the drop-down list, select the data field we wish to sort by **Ascending/Descending**
- Click **Next**
- Select a **Layout**
- Click **Next**
- Select a **Style**
- Click **Next**
- Type a **Title** for the **Report**
- Click **Finish**

3.5.2 Changing the Report design

Updating the report format

- Click **View**, **Design View** or click the view icon on the Standard Toolbar
- Click **Format**, **AutoFormat**
- Select a new **Format**
- Click **OK**
- Click **View**, **Print Preview** or click the view icon on the Standard Toolbar to return to the report

Changing the Report Title

- Click **View**, **Design View** or click the view icon on the Standard Toolbar
- Click on the box containing the report **Title**
- Type the **new Title** for the **Report**
- Click **View**, **Print Preview** or click the view icon on the Standard Toolbar to return to the report

3.5.3 Saving a Report

- Click **File**, **Save** or click the **Save icon** on the Standard Toolbar

3.5.4 Printing a Report

- Click **File**, **Print** or click the **print icon** on the Standard Toolbar

3.5.5 Closing a Report

- Click **File**, **Close**

3.6 Security

Access supports a rich array of security features to support the needs of different types of Access applications. Most multi-user Access applications can be benefit from user-level security, which lets developers designate groups of users. But some applications have more specialized needs.

3.6.1 Setting a database password

We can require users to enter a password to gain unrestricted Access to all Access data and database objects. Passwords are easy to administer compared to user level security. Password security is appropriate if we have a group whose members need equal access to all elements of a database file but not everyone in the office is a member of that group.

We can not use a password protected file as a member in a replica se because Jet database replication can not synchronize with a password protected file. We should also be careful about linking to database files with password protection because anyone who can access the file that links the protected file has unrestricted access to the protected file. Access stores an encrypted version of the password along with other information about the linked file. If someone changes the data in the database then the users will be unable to get the information according to the link of the data with other fields in other tables or queries. If someone changes the password for a linked file, Access prompts for the new password the next time another database file links to it.

To **assign and remove** a database **password**, we need exclusive access to the file. We can do that by following these steps:

- Open a file by choosing **Open Exclusive** from the open button in the Open dialog box to assign a password to a file
- Choose **Security-Set Database Password** from the Tools menu

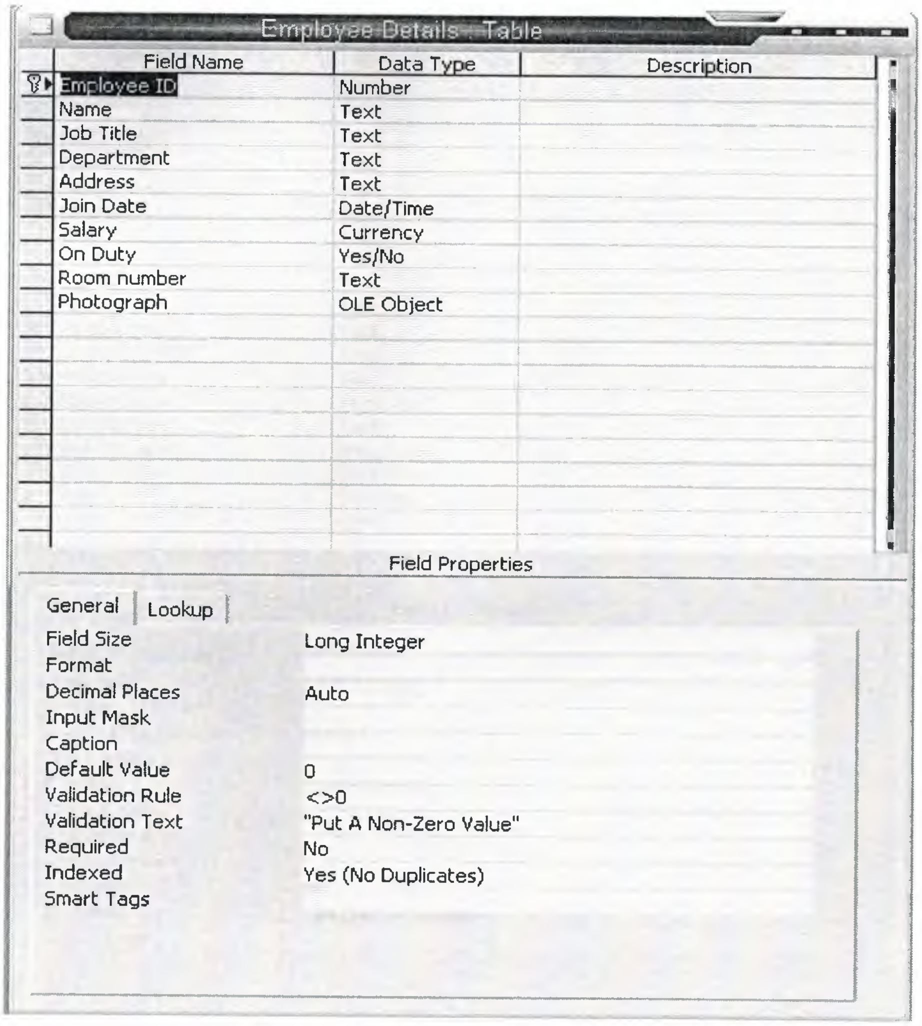
- In the **Set Database Password** dialog box, we enter the password of choice in the Password and verify text boxes and then click OK. The next time a user opens the file, the application will ask for the password
- After opening a database exclusively, choose **Tools-Security-Unset Database Password**. **Remove** the password by typing the password in the **Unset Database Password** dialog box. This removes the initial prompt for a password before a database is made available.

CHAPTER FOUR: HOTEL DATABASE DESIGN

In this chapter I described the works of a hotel and to describe the works in a easier method I created one database using the Microsoft Access. Here I have given the data tables, queries, forms, reports and the switchboard form.

4.1 Creating Tables for Hotel

Searching from some books I found that hotel's require some fixed fields in it. For every hotel these fields are fixed, the data in the fields may vary hotel to hotel. For that reason I started to create a table which is one of the main important part of a hotel, the 'Employee Details' table.



Employee Details : Table

Field Name	Data Type	Description
Employee ID	Number	
Name	Text	
Job Title	Text	
Department	Text	
Address	Text	
Join Date	Date/Time	
Salary	Currency	
On Duty	Yes/No	
Room number	Text	
Photograph	OLE Object	

Field Properties

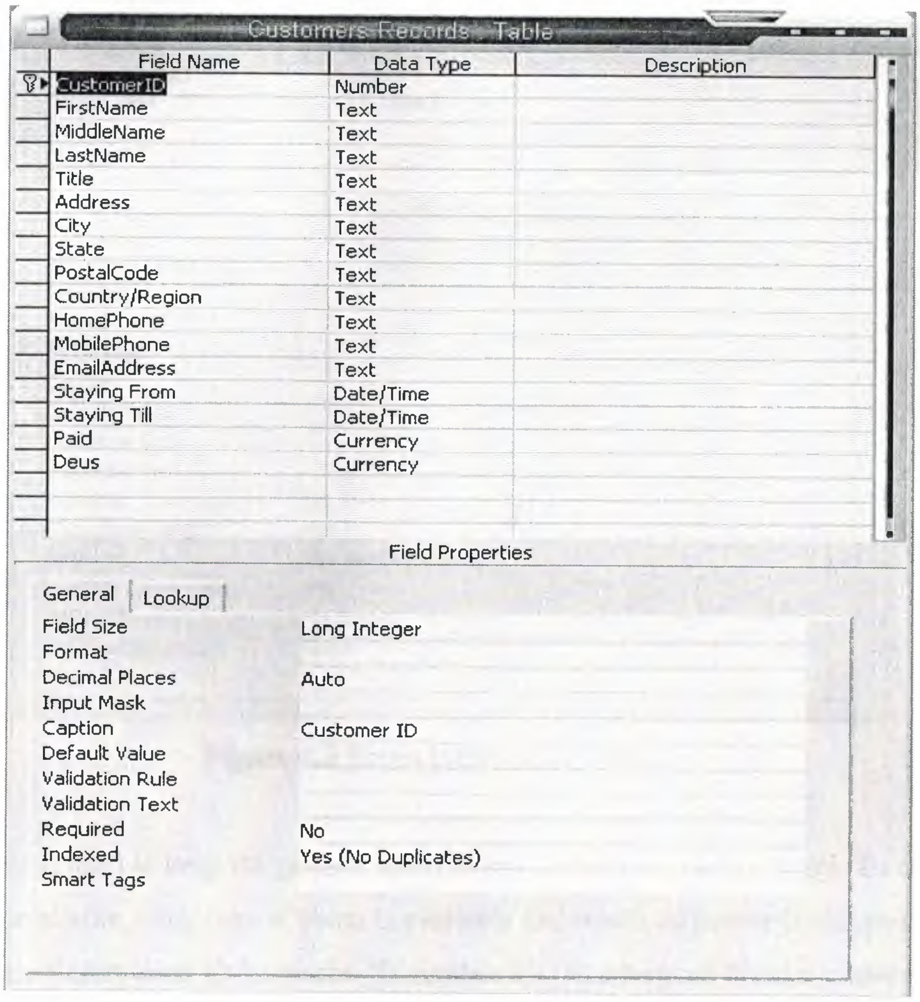
General | Lookup

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	0
Validation Rule	<>0
Validation Text	"Put A Non-Zero Value"
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	

Figure 4.1 Employee Details Table

In a hotel if we create a database for that hotel, we must gather the information of the employees. Here the 'Employee Details' table contains all necessary fields that we will try to keep the data of an employee as an administrator or a manager. We have to put the information of the employee, according to the field's necessity. I put the 'Employee ID' as the primary key field. The employee ID must not be same for different employees. To search for information of an employee, having an unique ID which is the primary key is much faster and safer.

Another main part of the hotel for which we have to create a table to keep all the information, the 'Customers Records' table. I create the 'Customers Records' table as shown in figure below:



Field Name	Data Type	Description
CustomerID	Number	
FirstName	Text	
MiddleName	Text	
LastName	Text	
Title	Text	
Address	Text	
City	Text	
State	Text	
PostalCode	Text	
Country/Region	Text	
HomePhone	Text	
MobilePhone	Text	
EmailAddress	Text	
Staying From	Date/Time	
Staying Till	Date/Time	
Paid	Currency	
Deus	Currency	

Field Properties

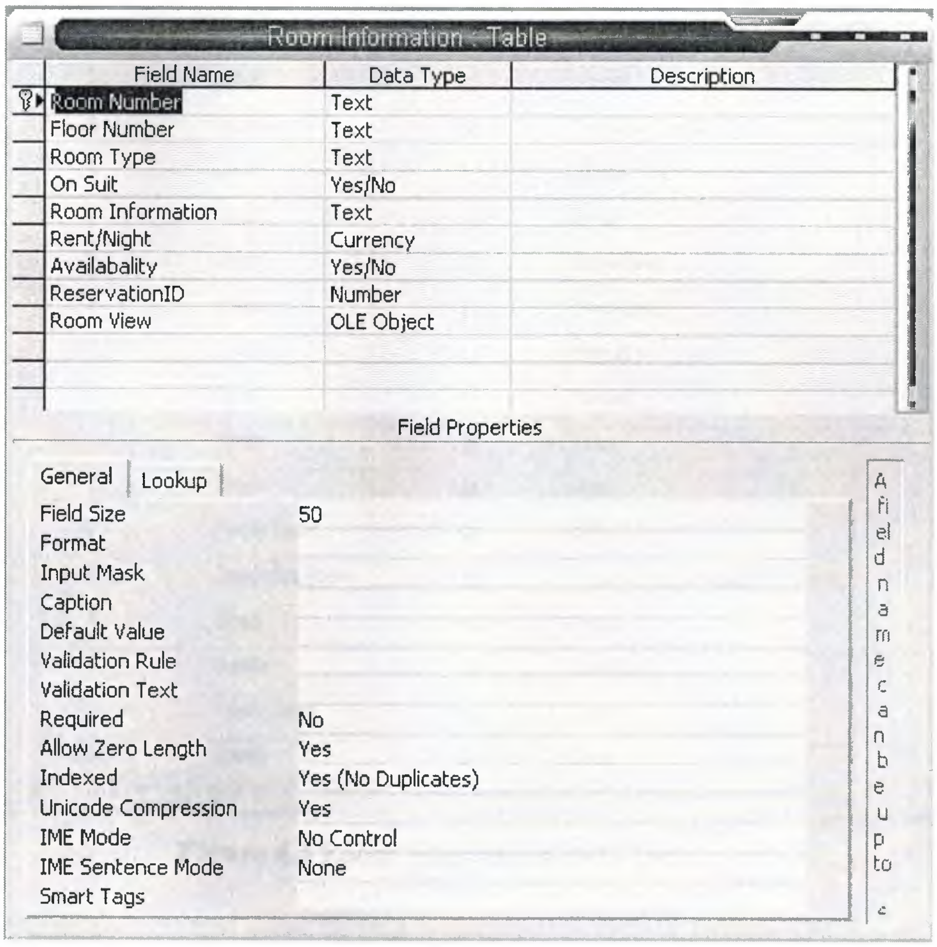
General | Lookup

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	Customer ID
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	

Figure 4.2 Customers Records Table

The table has created to keep all the information of customers. Each customer having a Customer ID which is the primary key of the table. So that we can get the accurate data of a customer. The table contains the duration the customer is staying in the hotel and the payment terms.

The next table I created is the 'Room Information' table, which has shown in the figure below:



Room Information : Table

Field Name	Data Type	Description
Room Number	Text	
Floor Number	Text	
Room Type	Text	
On Suit	Yes/No	
Room Information	Text	
Rent/Night	Currency	
Availabality	Yes/No	
ReservationID	Number	
Room View	OLE Object	

Field Properties

General | Lookup

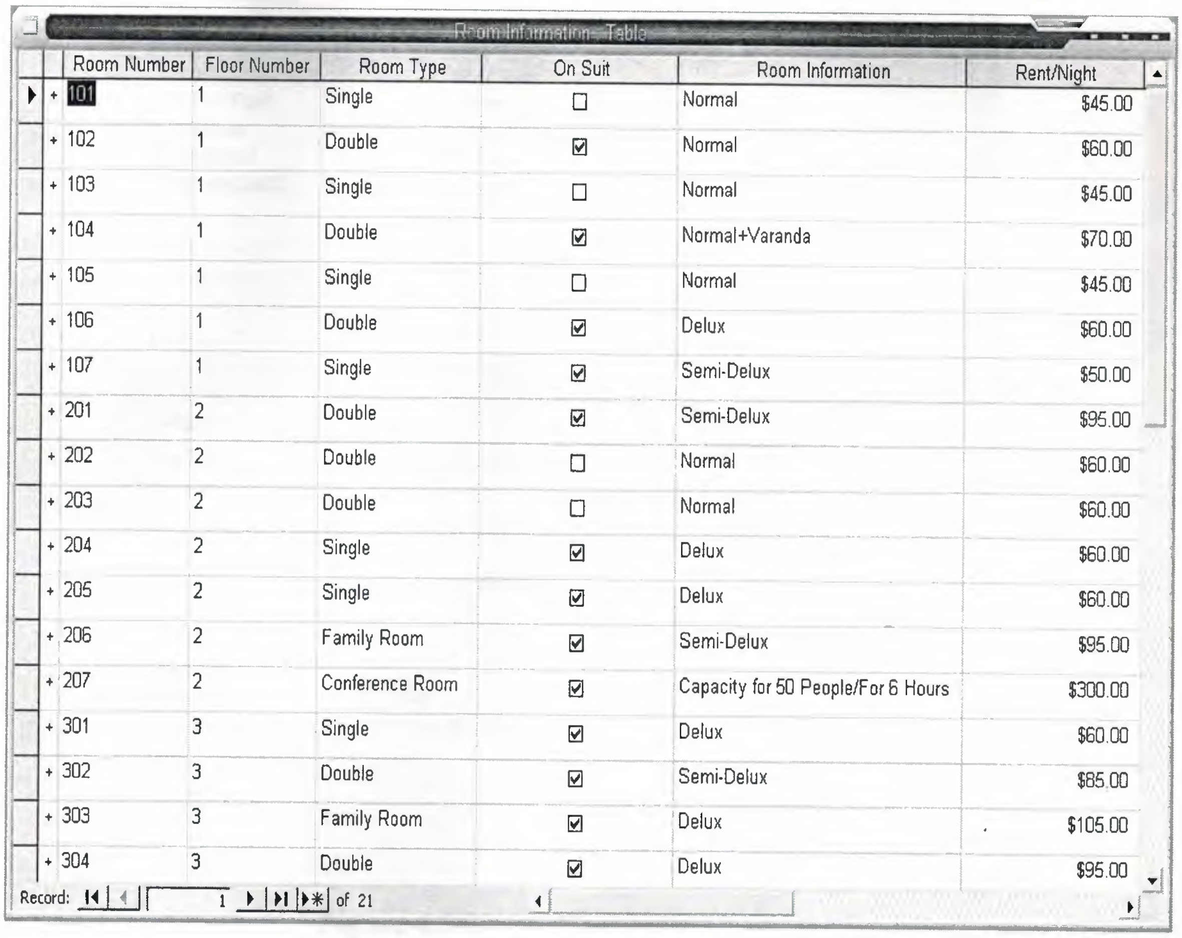
Field Size	50
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	Yes (No Duplicates)
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Smart Tags	

A field name can be up to

Figure 4.3 Room Information Table

This table is used to keep the general information of the rooms of the hotel. To check which room is available, what type of room is available and which customer is occupying which room. The Reservation ID keeps the ID number for the advanced booked customers that will keep reserved for the customer for the date it has reserved from and reserved till. By letting know the ID number we can easily find that which room has been booked for that customer.

The data table will look like the figure I have given below:

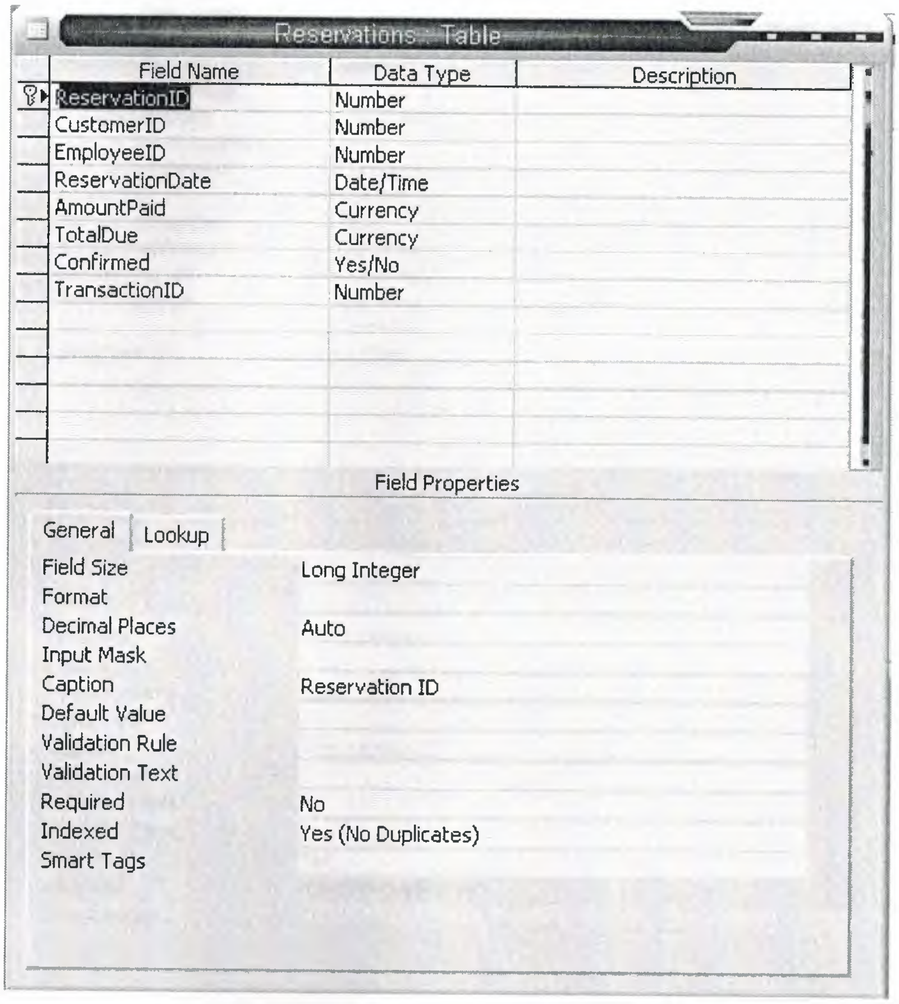


Room Number	Floor Number	Room Type	On Suit	Room Information	Rent/Night
101	1	Single	☐	Normal	$45.00
102	1	Double	☑	Normal	$60.00
103	1	Single	☐	Normal	$45.00
104	1	Double	☑	Normal+Varanda	$70.00
105	1	Single	☐	Normal	$45.00
106	1	Double	☑	Delux	$60.00
107	1	Single	☑	Semi-Delux	$50.00
201	2	Double	☑	Semi-Delux	$95.00
202	2	Double	☐	Normal	$60.00
203	2	Double	☐	Normal	$60.00
204	2	Single	☑	Delux	$60.00
205	2	Single	☑	Delux	$60.00
206	2	Family Room	☑	Semi-Delux	$95.00
207	2	Conference Room	☑	Capacity for 50 People/For 6 Hours	$300.00
301	3	Single	☑	Delux	$60.00
302	3	Double	☑	Semi-Delux	$85.00
303	3	Family Room	☑	Delux	$105.00
304	3	Double	☑	Delux	$95.00

Record: 1 of 21

Figure 4.4 Room Information Data Table

The next table I created is the 'Reservations' table. The fields section has shown in the figure below:


Reservations : Table

Field Name	Data Type	Description
ReservationID	Number	
CustomerID	Number	
EmployeeID	Number	
ReservationDate	Date/Time	
AmountPaid	Currency	
TotalDue	Currency	
Confirmed	Yes/No	
TransactionID	Number	

Field Properties

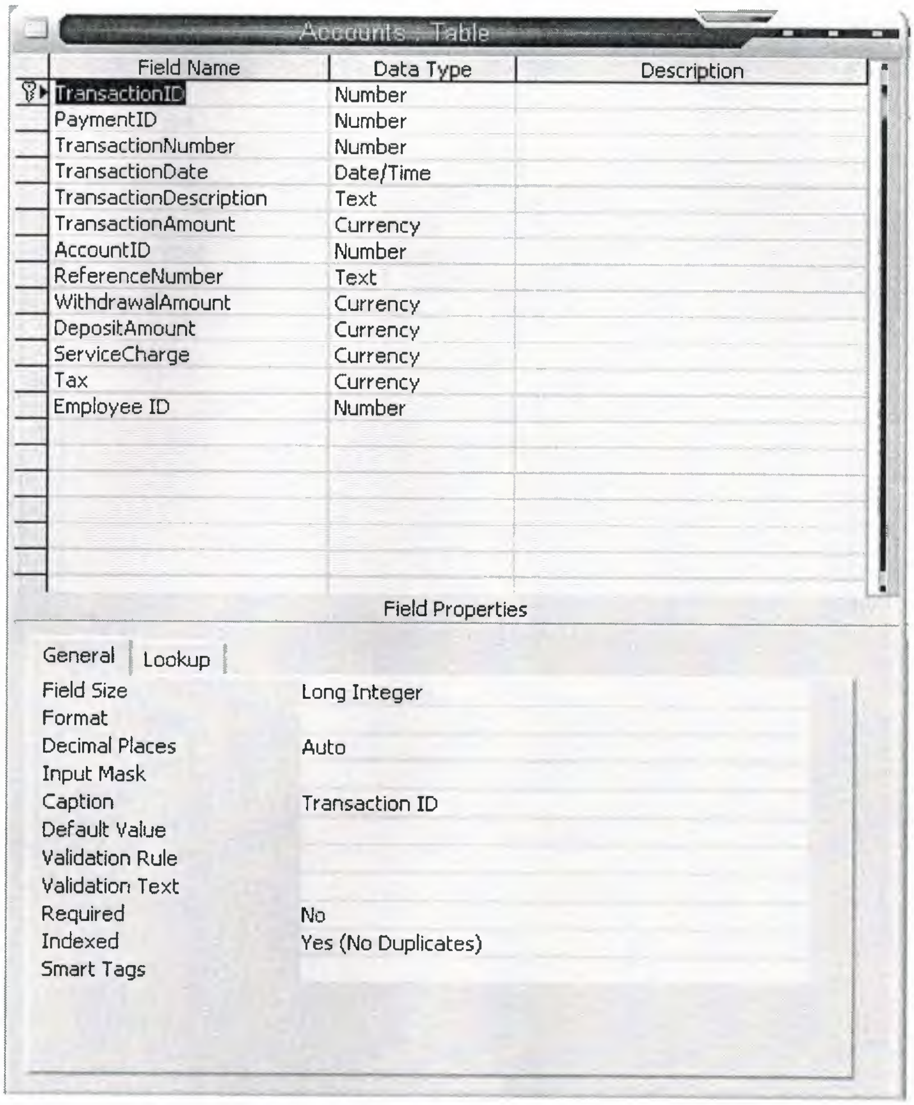
General | Lookup

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	Reservation ID
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	

Figure 4.5 Reservations table

This table has been created to let the customers to book a room few days earlier of his coming, so that he may get a room of the hotel definitely. ReservationID is the primary key here, to keep the data unique for the relationships between the tables. This table contains the customer ID, employee ID, reservation date, amount paid in advanced, room is confirmed for the customer or not, and the transaction ID to keep the information of the payment the customer has made in the account section .

The next table I created is the 'Accounts' table. The fields section of the table has shown below:


Accounts : Table

Field Name	Data Type	Description
TransactionID	Number	
PaymentID	Number	
TransactionNumber	Number	
TransactionDate	Date/Time	
TransactionDescription	Text	
TransactionAmount	Currency	
AccountID	Number	
ReferenceNumber	Text	
WithdrawalAmount	Currency	
DepositAmount	Currency	
ServiceCharge	Currency	
Tax	Currency	
Employee ID	Number	

Field Properties

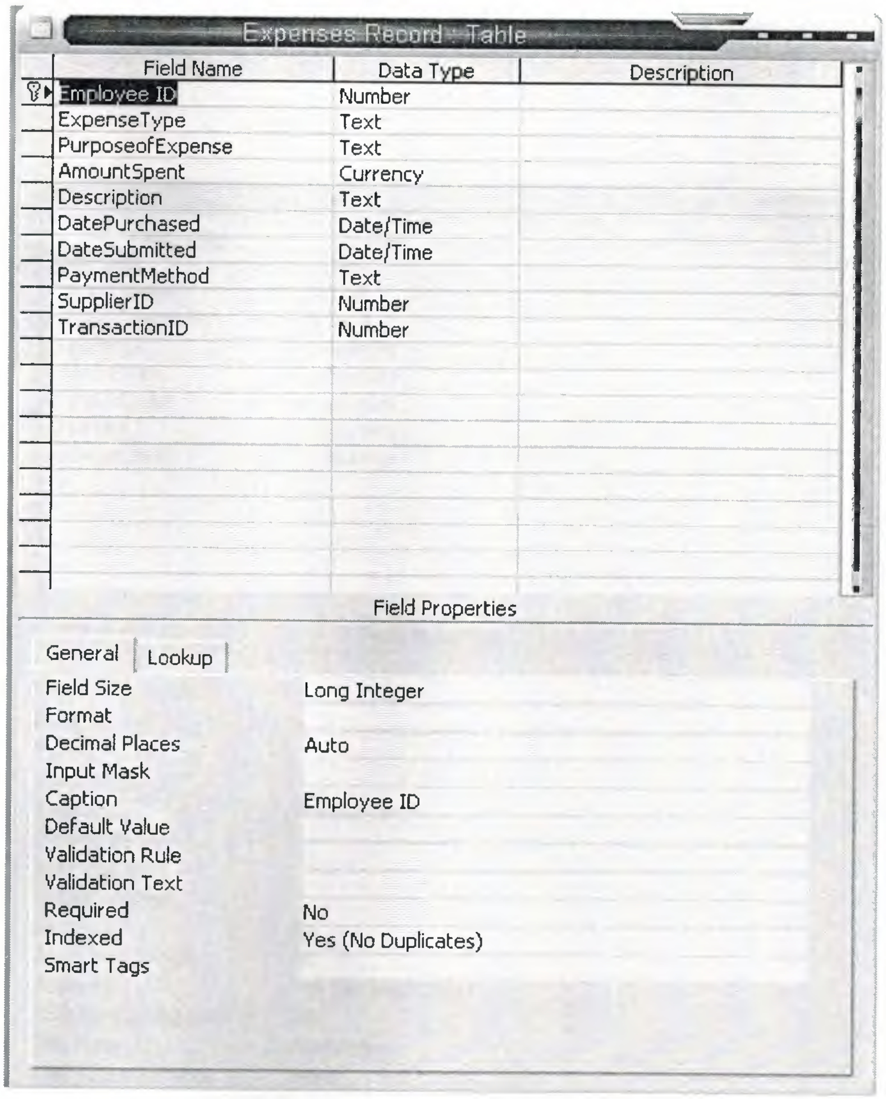
General | Lookup

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	Transaction ID
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	

Figure 4.6 Accounts Table

This table keeps all the information of the monitory matters of the hotel. Here I just maid one primary key which is the TransactionID. I can make two primary keys which will make it composite key. But I chose the TransactionID as unique for this table where the other fields like PaymentID, AccountID or the EmployeeID will work here as the foreign key fields. To keep all the information of money getting from the customers or expenses done for the hotel, this table will keep these information uniquely for all fields related to it.

The next table I created is the 'Expenses Record' table. The fields section of the table has shown below:



Expenses Record : Table

Field Name	Data Type	Description
Employee ID	Number	
ExpenseType	Text	
PurposeofExpense	Text	
AmountSpent	Currency	
Description	Text	
DatePurchased	Date/Time	
DateSubmitted	Date/Time	
PaymentMethod	Text	
SupplierID	Number	
TransactionID	Number	

Field Properties

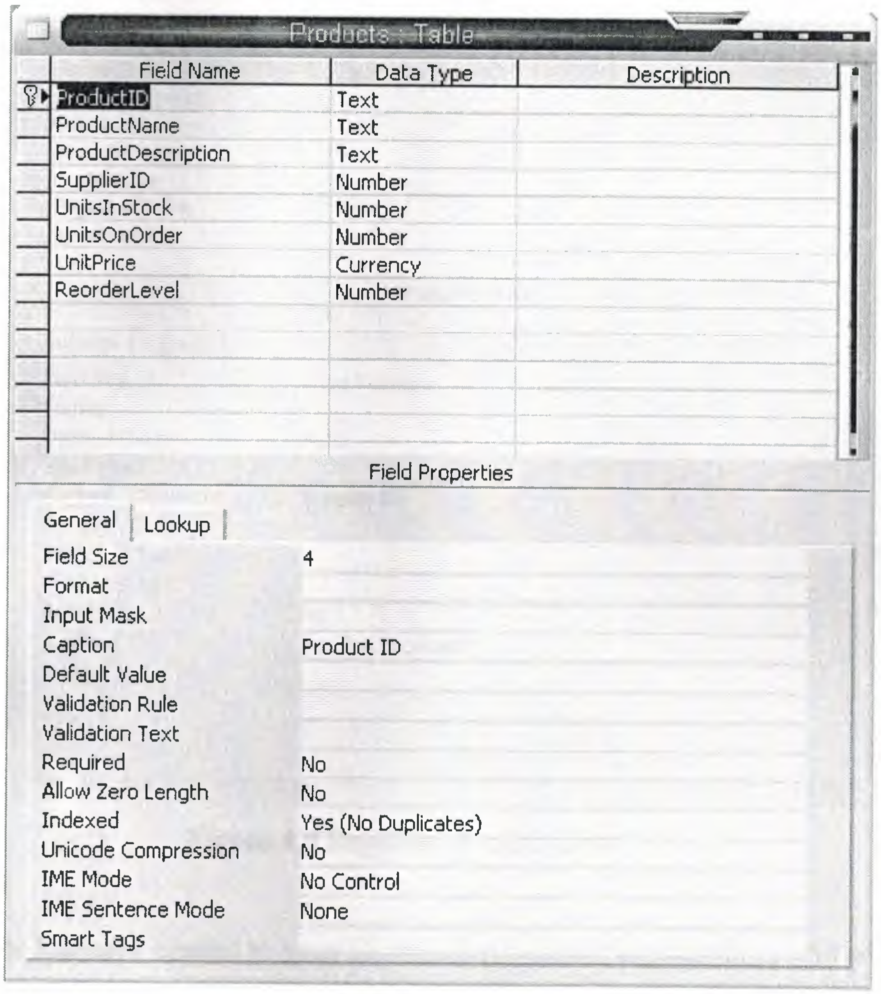
General | Lookup

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	Employee ID
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	

Figure 4.7 Expenses Record Table

In this table we can keep all the information of the expenses has been made for the hotel. Here I made the EmployeeID as the primary key as the expenses will definitely be made by an employee and it can be easily saved here under the employees ID. Here the expenses type keeps the information of the item the expenses has made for; purpose of expenses keeps the information of the reason for the expense has been made. The other fields keep information of the purchased date, the payment method, and the transaction ID to keep the amount of expenses record in the 'Accounts' table.

The next table I created is the 'Products' table. The fields section of the table has shown below:


Products : Table

Field Name	Data Type	Description
ProductID	Text	
ProductName	Text	
ProductDescription	Text	
SupplierID	Number	
UnitsInStock	Number	
UnitsOnOrder	Number	
UnitPrice	Currency	
ReorderLevel	Number	

Field Properties

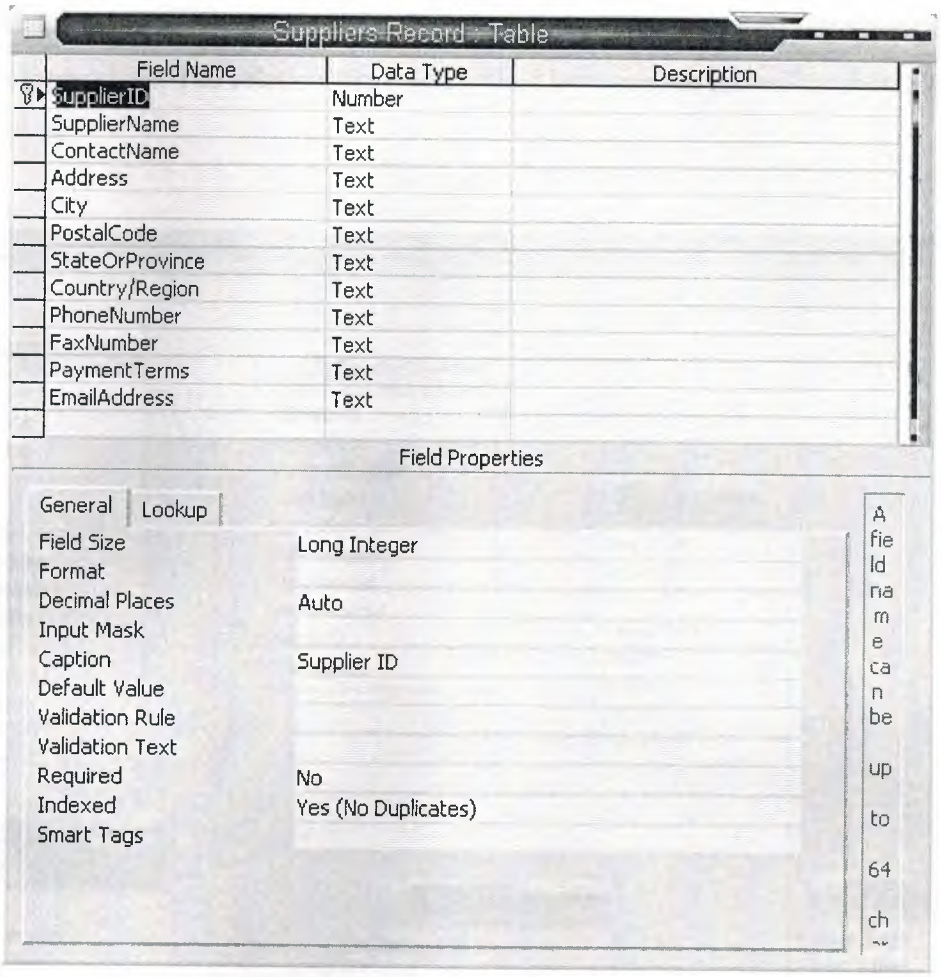
General | Lookup

Field Size	4
Format	
Input Mask	
Caption	Product ID
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	No
Indexed	Yes (No Duplicates)
Unicode Compression	No
IME Mode	No Control
IME Sentence Mode	None
Smart Tags	

Figure 4.8 Products Table

I have been created this table to keep all the information of the products has been bought for the hotel. This will keep the data of the product bought, the description of the product, supplier ID, unit price and the reorder level by which we can know the importance of the order of this product.

The next table I created is the 'Suppliers Record' table. And the fields section of this table has shown below:


Suppliers Record : Table

Field Name	Data Type	Description
SupplierID	Number	
SupplierName	Text	
ContactName	Text	
Address	Text	
City	Text	
PostalCode	Text	
StateOrProvince	Text	
Country/Region	Text	
PhoneNumber	Text	
FaxNumber	Text	
PaymentTerms	Text	
EmailAddress	Text	

Field Properties

General | Lookup

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	Supplier ID
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	

A field name can be up to 64 ch

Figure 4.9 Suppliers Record Table

This table has been created to keep all the information of the suppliers who will supply different types of product necessary for the hotel. The suppliers will provide all the goods need for the hotel, for that reason it's very important to keep the information of the suppliers so that we can order the necessary product immediately.

4.2 Constructing relationships between tables

For getting the information quickly and exactly I need to create relationships between the tables. The relationships will allow the tables to share all the data required for all the tables between themselves. The relationship I established between the tables has shown in the figure below:

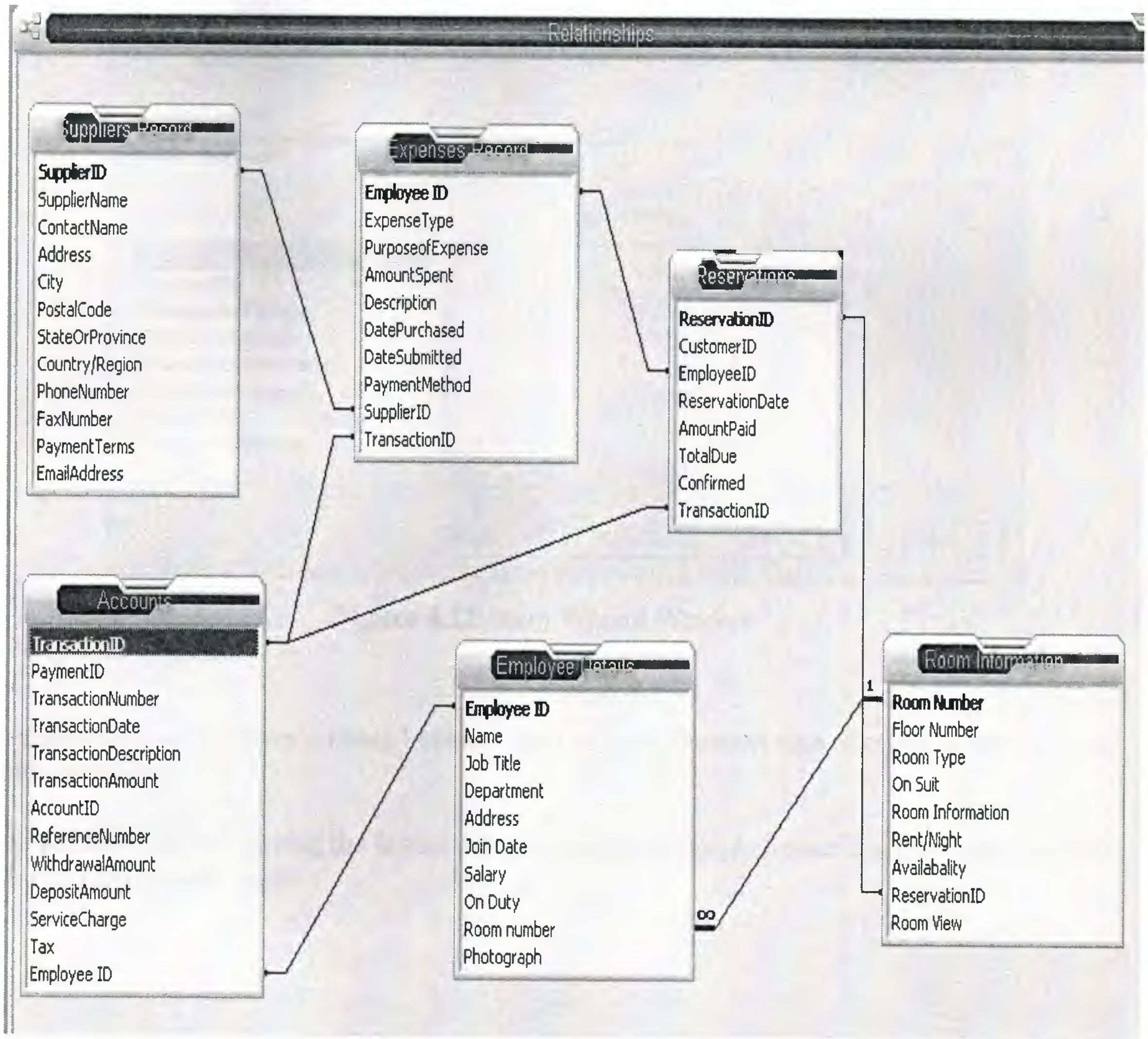
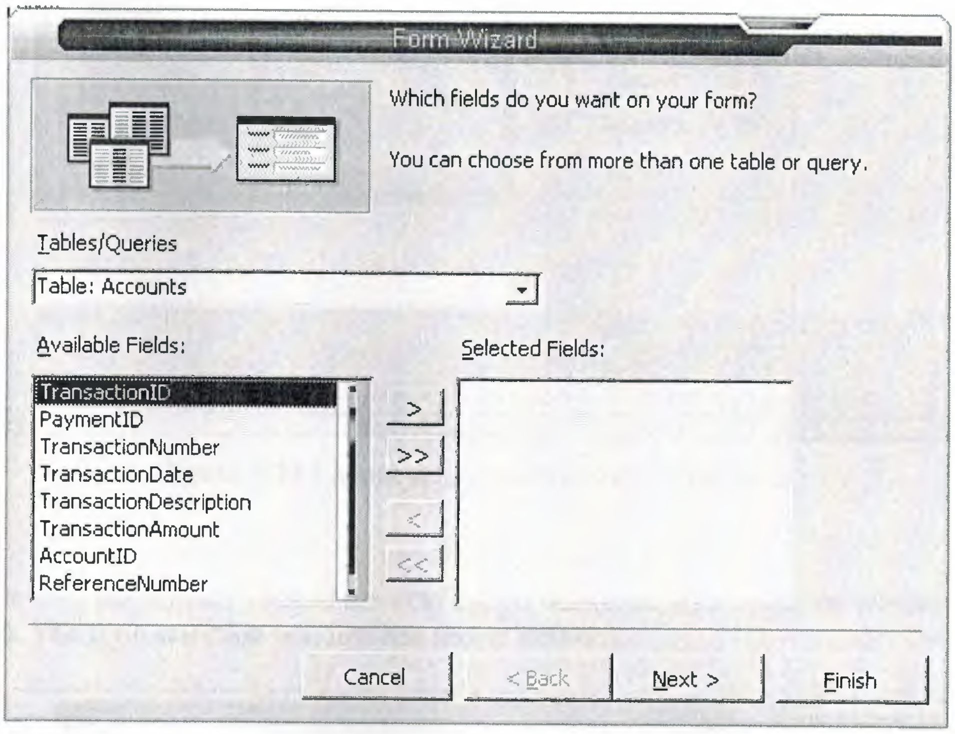


Figure 4.10 Structure of relationship between the tables

4.3 Creating Forms for my application

I created the forms by using the form wizard. The wizard made the job very easy while creating the forms. There are all the options given for the forms type I want to see in my form. I choose the wizard to create the form for the 'Accounts'. I selected the fields I need for this type of form by clicking the single arrow shown in the figure below:



Which fields do you want on your form?

You can choose from more than one table or query.

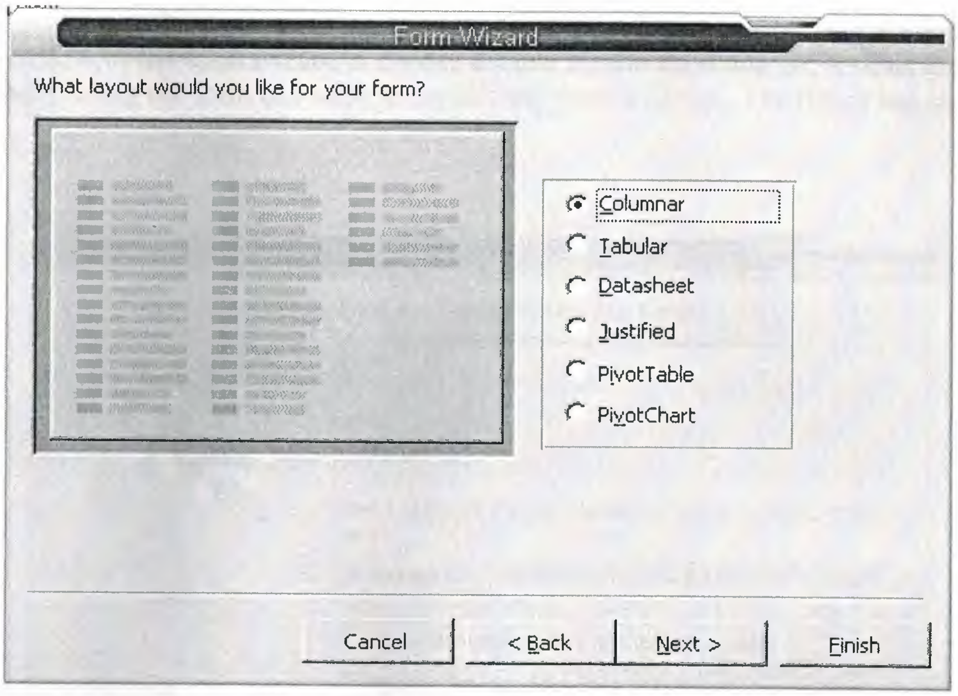
Table: Accounts

TransactionID
PaymentID
TransactionNumber
TransactionDate
TransactionDescription
TransactionAmount
AccountID
ReferenceNumber

Figure 4.11 Form Wizard Window

After selecting the form's fields I clicked next to go to the next step of creating the form for the accounts.

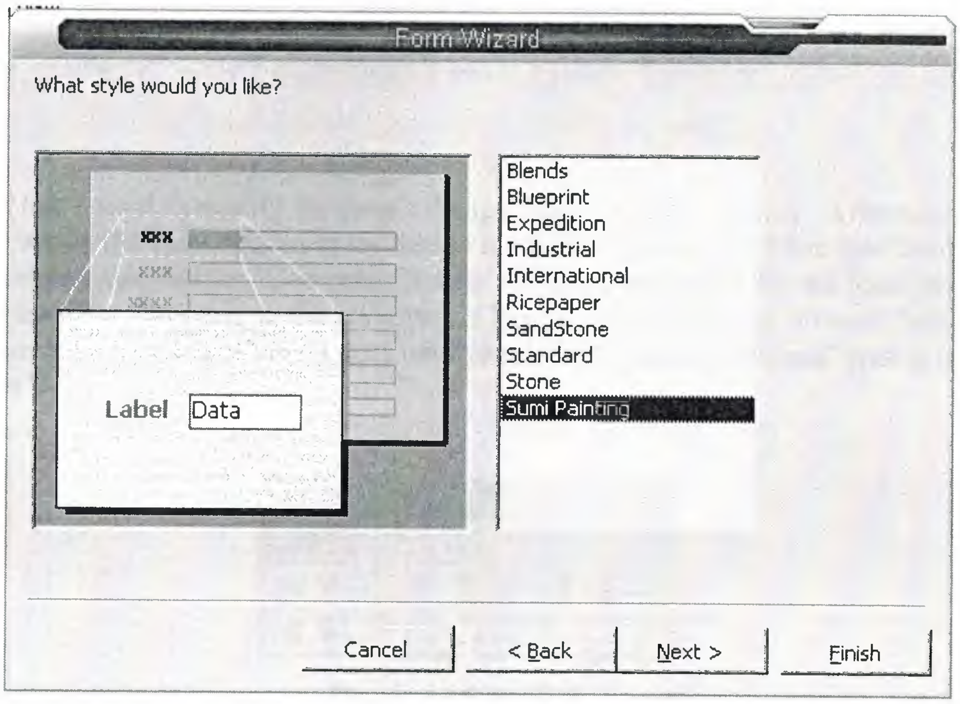
In the next step it's giving the layout selection for the form. I choose the 'Columnar' layout for my 'Accounts' form.


What layout would you like for your form?

[✓ Columnar] Tabular Datasheet Justified PivotTable PivotChart

Figure 4.12 Layout selection window for the form

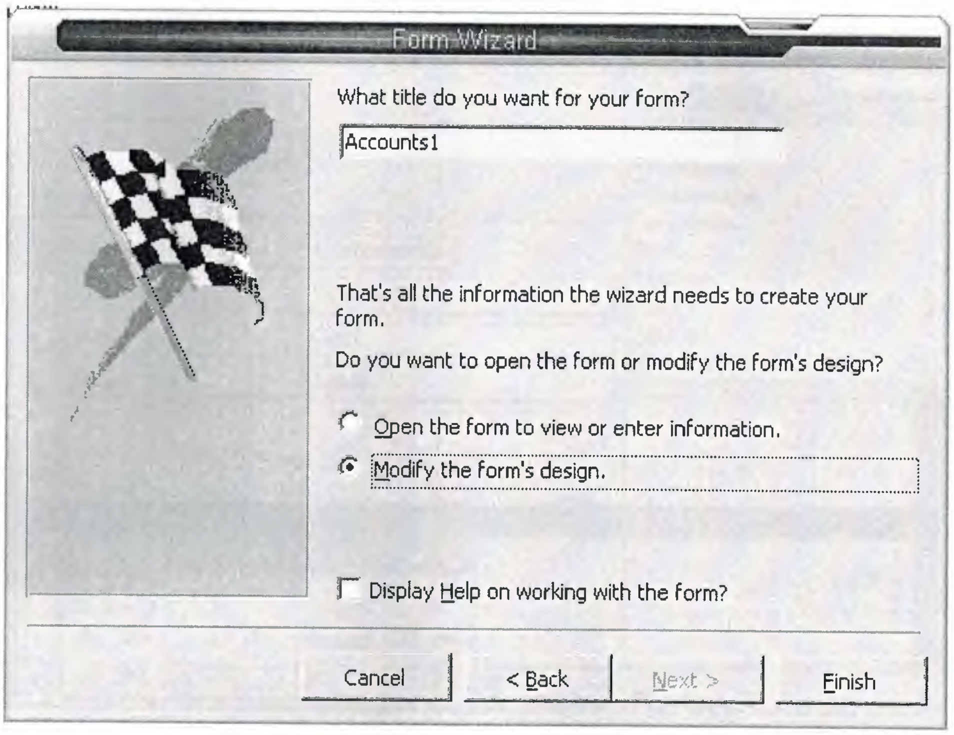
After selecting the layout I clicked the next button where the style selection window appeared. The style selection window has shown below:


What style would you like?

Blends
Blueprint
Expedition
Industrial
International
Ricepaper
SandStone
Standard
Stone
[✓ Sumi Painting]

Figure 4.13 Style selection window for the form

I choose the 'Sumi Painting' style for my form. After that I clicked next, where I found the 'Title' window of the form wizard to choose a name for the form and the options to enter the data directly using the form or I wish to modify the form's design. The figure has shown below:



What title do you want for your form?

Accounts1

That's all the information the wizard needs to create your form.

Do you want to open the form or modify the form's design?

Open the form to view or enter information. [✓ Modify the form's design.]

Display Help on working with the form?

Figure 4.14 Final window of the form wizard

I choose that I want to modify the form's design. Then I clicked 'Finish'. After clicking the 'Finish' button the form opened in the design mode like figure 4.16. From the 'Toolbox' shown below I selected the 'Command Button' to create one button for the form which will work in the form according to the command. I have to be sure that the 'Control Wizard' option has been selected or not; if not I have to select the 'Control Wizard' option from the 'Toolbox'.

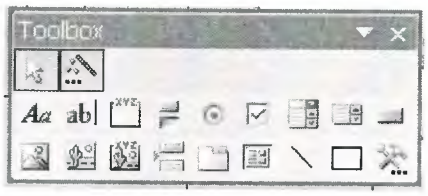


Figure 4.15 Toolbox

It is important to select the form's 'Header' and 'Footer' section. In the 'Footer' section we have to put the 'Command Button' to let the 'Control Wizard' to work. Otherwise the 'Control Wizard' will not work for the 'Command Button'.

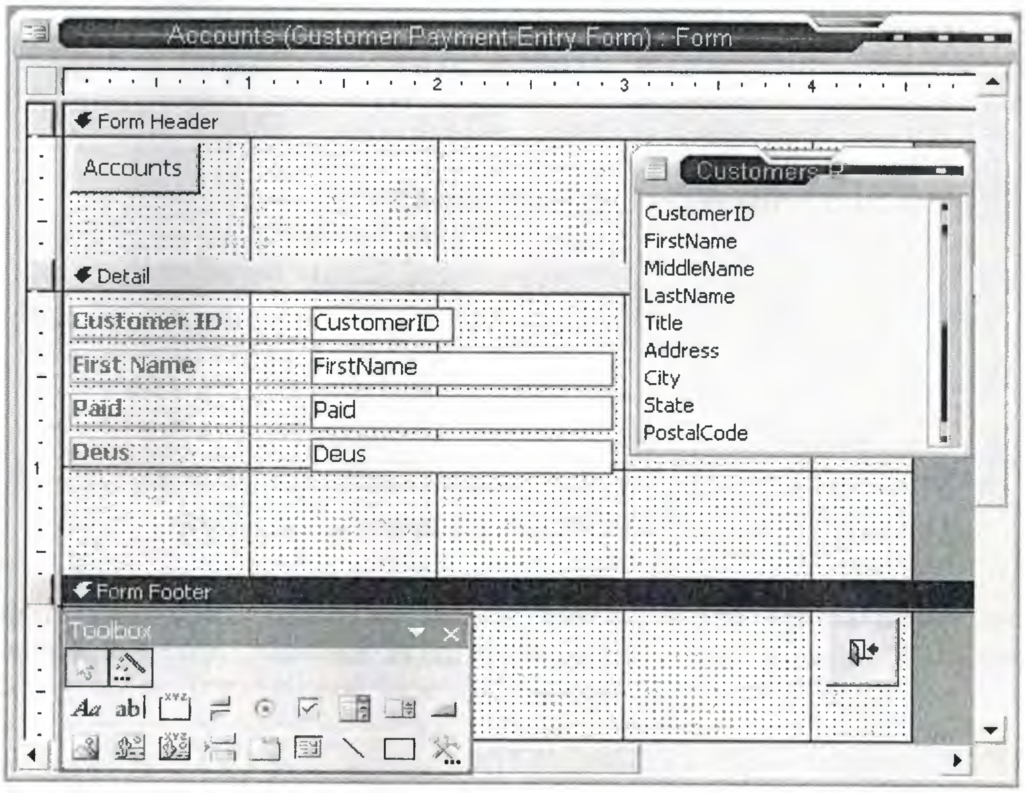
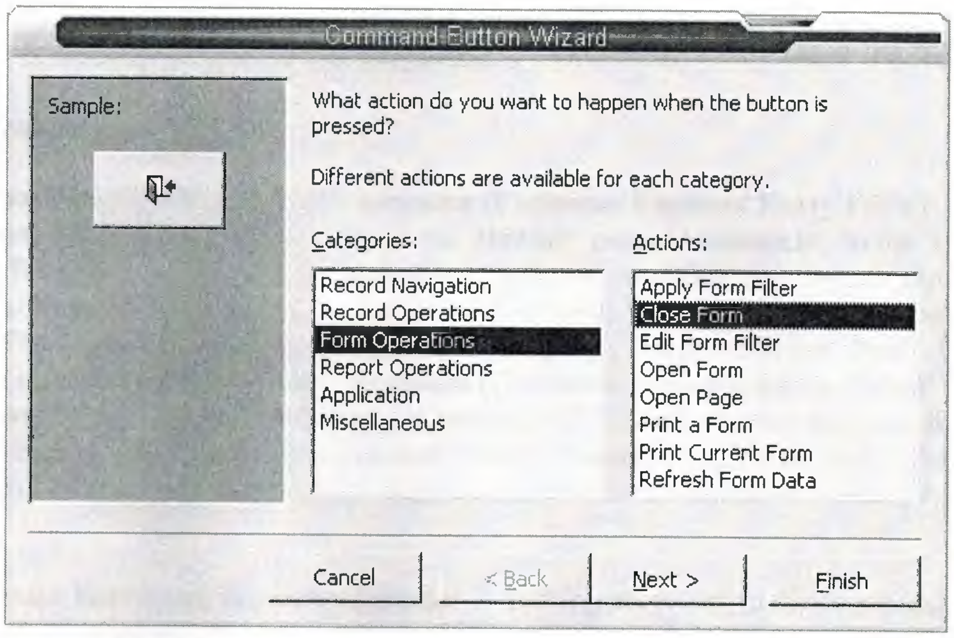


Figure 4.16 Modifying the Form's Design

When I put the button in the 'Footer' section the 'Command Button Wizard' started. The wizard has shown in the figure in the next page (Figure 4.17).


Command Button Wizard

Sample:

What action do you want to happen when the button is pressed?

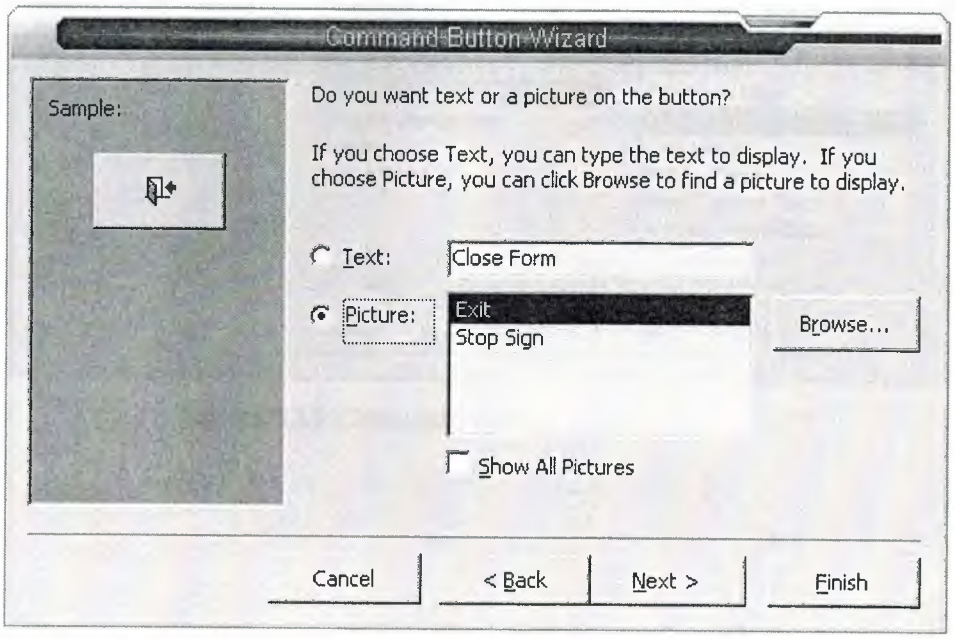
Different actions are available for each category.

Categories:
- Record Navigation
- Record Operations
- Form Operations
- Report Operations
- Application
- Miscellaneous

Actions:
- Apply Form Filter
- Close Form
- Edit Form Filter
- Open Form
- Open Page
- Print a Form
- Print Current Form
- Refresh Form Data

Figure 4.17 Command Button Wizard Window

There are six 'Categories', where in each category there are several 'Actions' options are given. I choose the 'Form Operations' in the categories section and choose the 'Close Form' in the actions section. Then I clicked 'Next'. The window shown in the figure below appeared:


Command Button Wizard

Sample:

Do you want text or a picture on the button?

If you choose Text, you can type the text to display. If you choose Picture, you can click Browse to find a picture to display.

Text: Close Form

[✓ Picture:]
- Exit
- Stop Sign

Show All Pictures

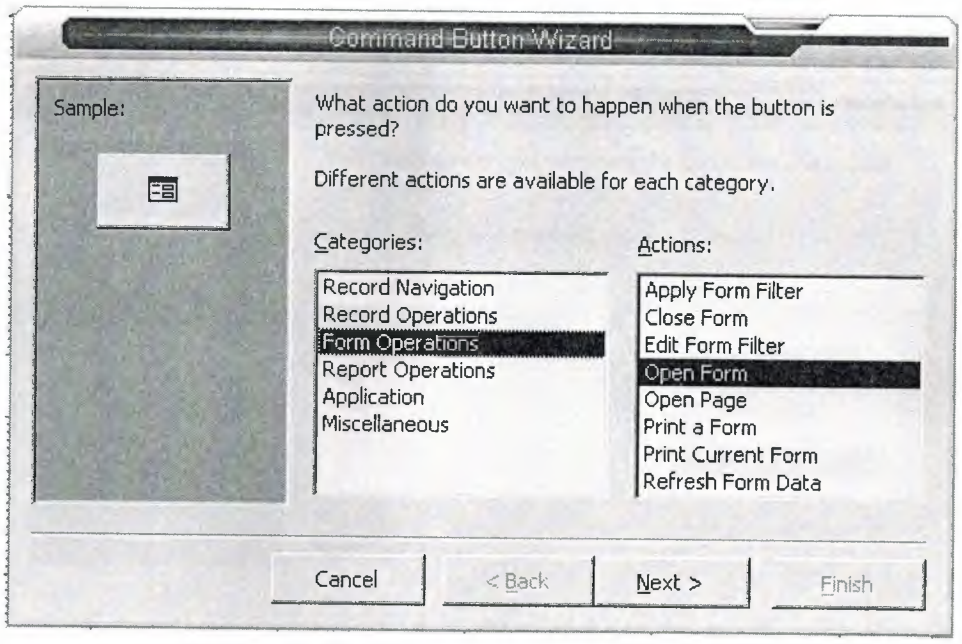
Figure 4.18 Next step of the Command Button Wizard

I choose the 'Picture' section and selected the 'Exit' sign for the button I created. Then I finished the wizard clicking the 'Finish' button.

4.3.1 Creating Pop-Up Form

I made one '**Pop-Up Form**' in the '**Accounts (Customer Payment Entry Form)**' form by adding one '**Command Button**' in the form '**Header**' named '**Accounts**'. In the '**Command Button Wizard**' I choose '**Form Operations**' in the '**Categories**' and choose '**Open Form**' in the '**Actions**'. After selecting the form I want to see as '**Pop-Up**' I clicked '**Next**'. Clicked '**Open The Form And Find Specific Data To Display**', and then clicked '**Next**'. In the wizard I created the links between '**Accounts (Customer Payment Entry Form)**' form's '**CustomerID**' with '**Accounts**' form's '**CustomerID**'. Then I selected the name for the button which is '**Accounts**'. Then I clicked '**Finish**' button to end the process. All the processes I did to create the 'Pop-Up' form have shown in the figures 4.19, 4.20, 4.21 and 4.22.

I opened the form to see the button I created is working according to the command or not. I found that it's working properly.

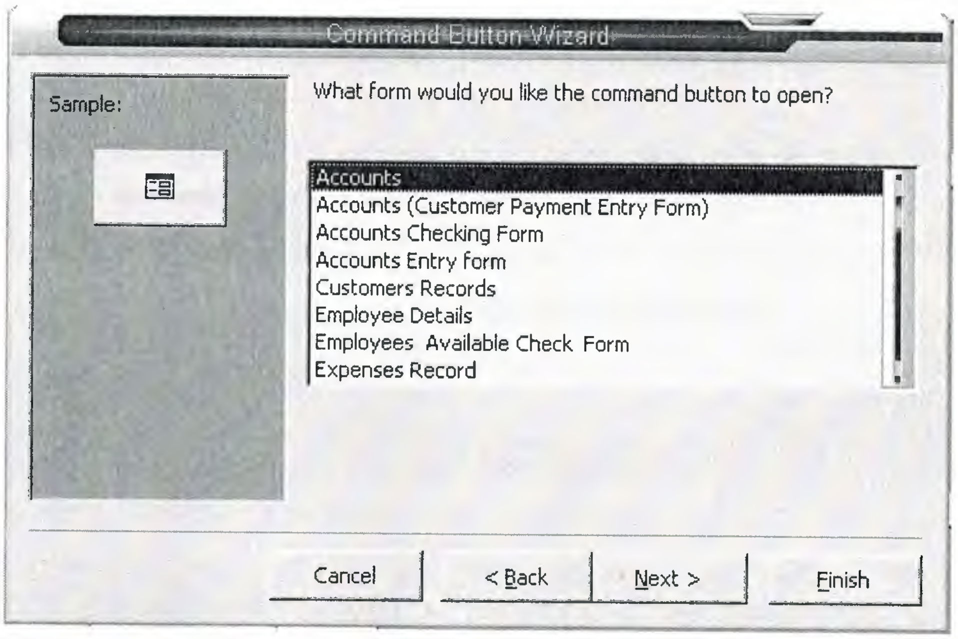


Command Button Wizard

What action do you want to happen when the button is pressed?

Different actions are available for each category.

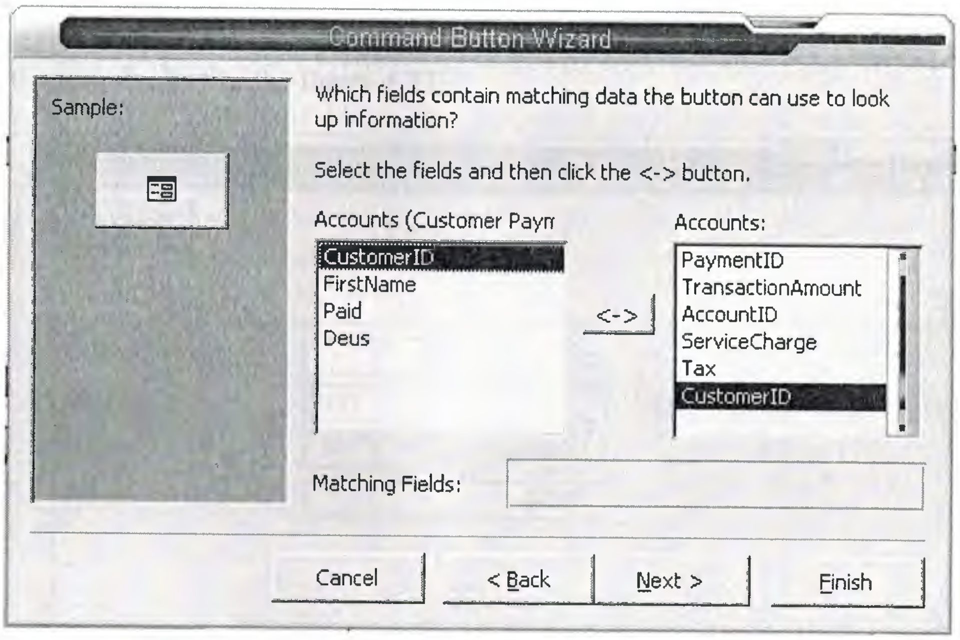
Figure 4.19 Command Button Wizard



What form would you like the command button to open?

Accounts
Accounts (Customer Payment Entry Form)
Accounts Checking Form
Accounts Entry form
Customers Records
Employee Details
Employees Available Check Form
Expenses Record

Figure 4.20 Command Button Wizard to Select Form



Which fields contain matching data the button can use to look up information?

Select the fields and then click the <-> button.

Accounts (Customer Paym

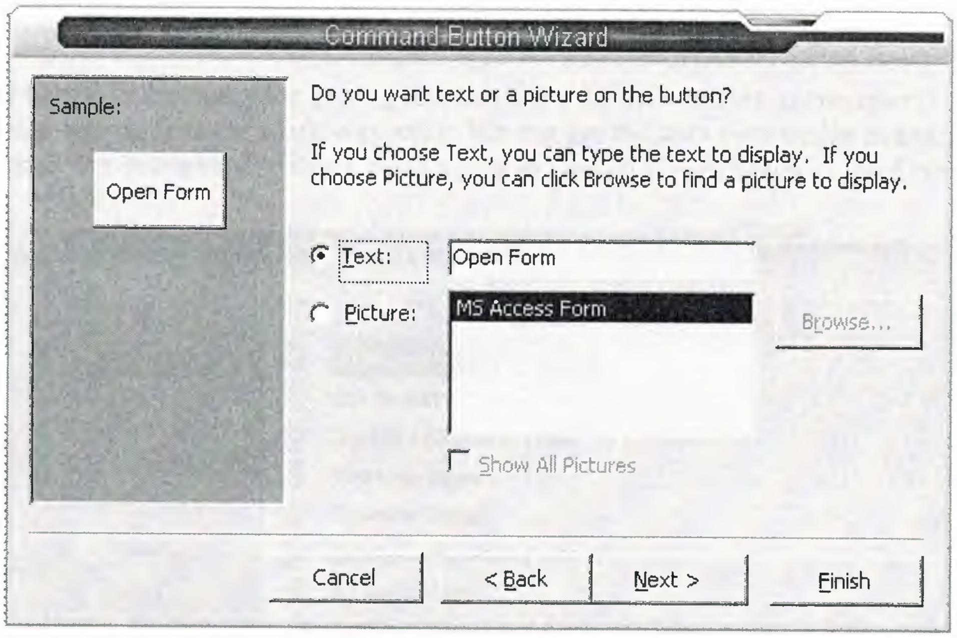
CustomerID
FirstName
Paid
Deus

Accounts:

PaymentID
TransactionAmount
AccountID
ServiceCharge
Tax
CustomerID

Matching Fields:

Figure 4.21 Creating Links Between Two Forms


Command Button Wizard

Sample:

Open Form

Do you want text or a picture on the button?

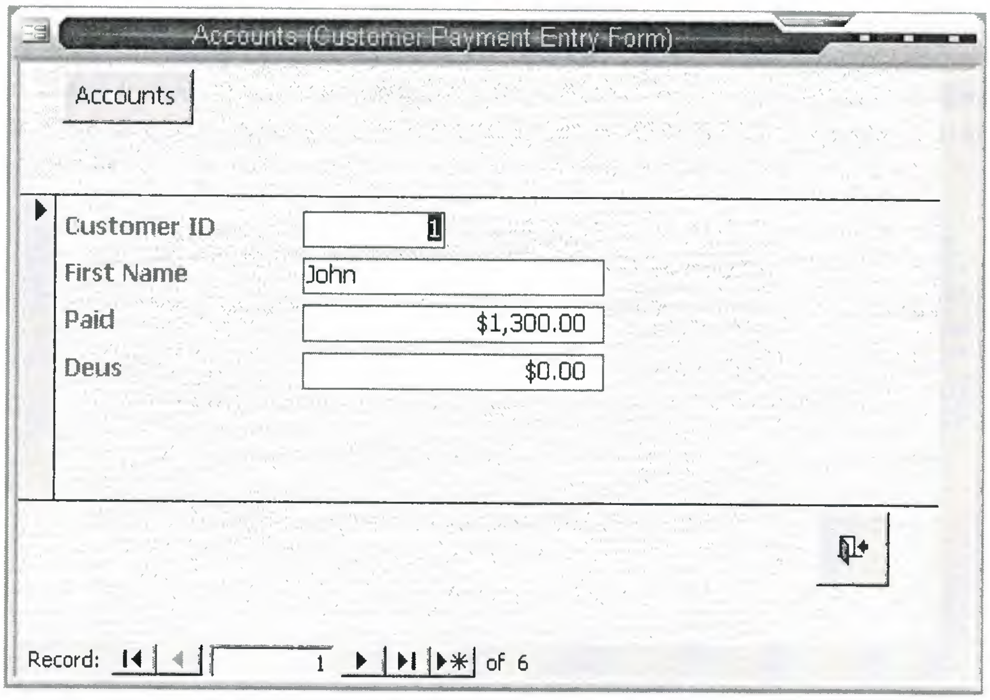
If you choose Text, you can type the text to display. If you choose Picture, you can click Browse to find a picture to display.

[✓ Text:] Open Form

Picture: MS Access Form

Figure 4.22 Giving a Name to the Button

After creating these two 'Command Button's in the 'Accounts (Customer Payment Entry Form)', the table looks like the figure 4.23.


Accounts (Customer Payment Entry Form)

Accounts

Customer ID	1
First Name	John
Paid	$1,300.00
Deus	$0.00

Record: 1 of 6

Figure 2.23 View of the Accounts Form

4.4 Creating SQL Operation for Hotel

We need to create the query for getting the data from different tables in one query application. This makes the work very easy. We can get the data very easily and quickly. I have created the queries using the 'Create query in Design view' shown in the figure below:

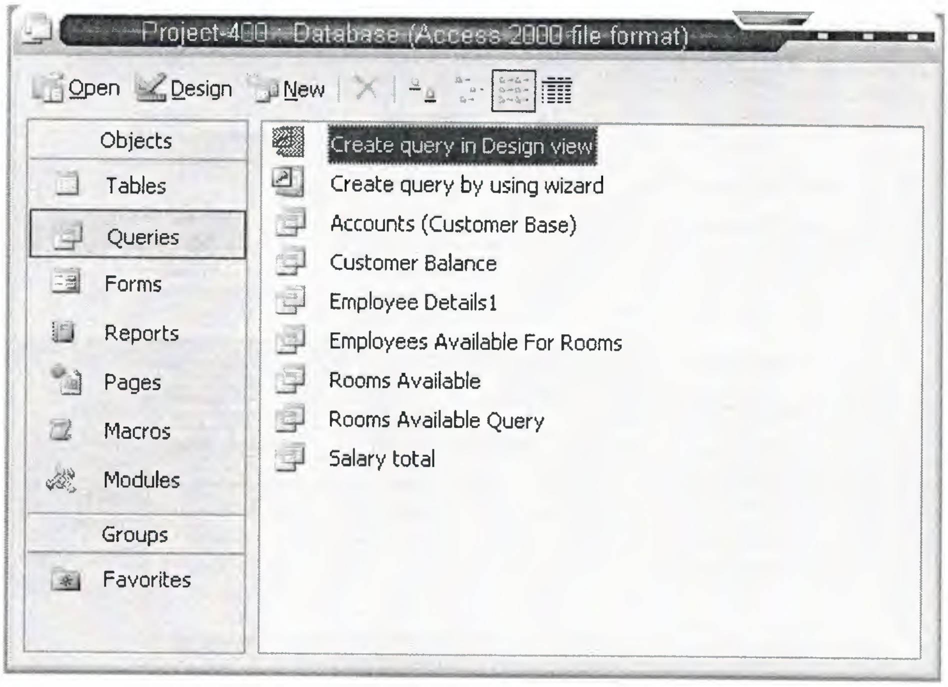


Figure 4.24 Creating query in Design view

I choose 'Room Information' from the tables and add it to the query, then choose 'Rooms Available' from the forms and add it to the query. The window of 'Show Table' has shown in the figure below:

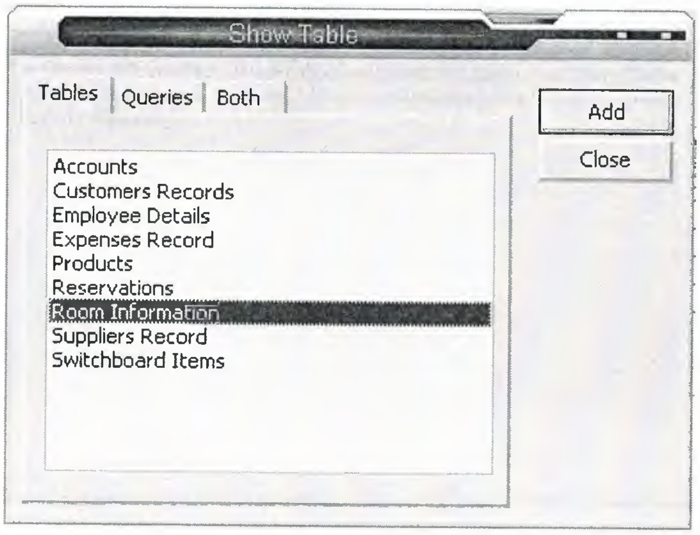


Figure 4.25 Show Table Window

I closed the 'Show Table' window and clicked right button on the query (Figure 4.26). I selected the 'SQL View' to see the structured query language window to write the SQL command for the 'Rooms Available' linked with the 'Room Information' to get the appropriate data.

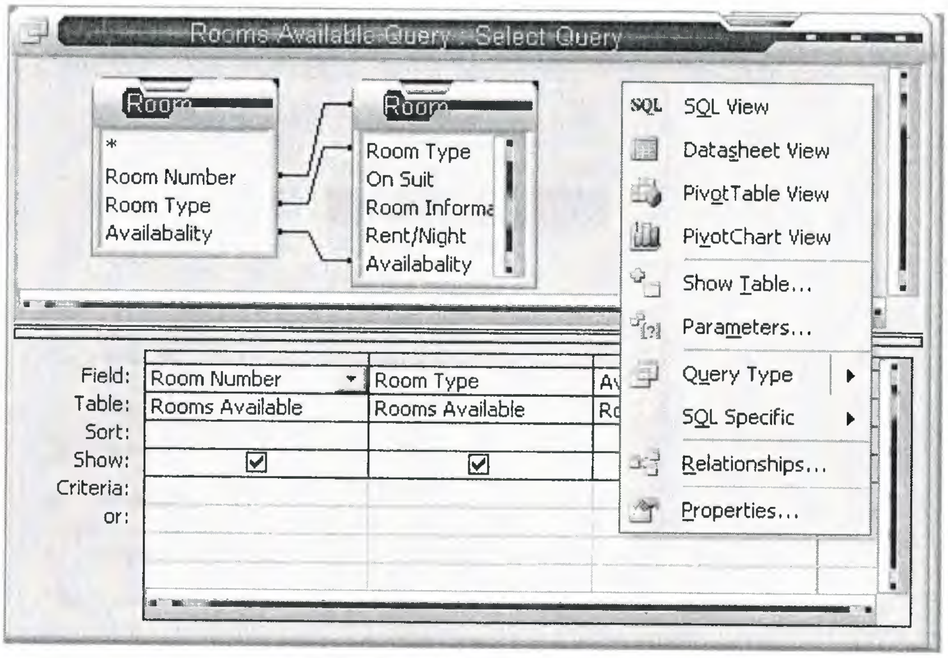
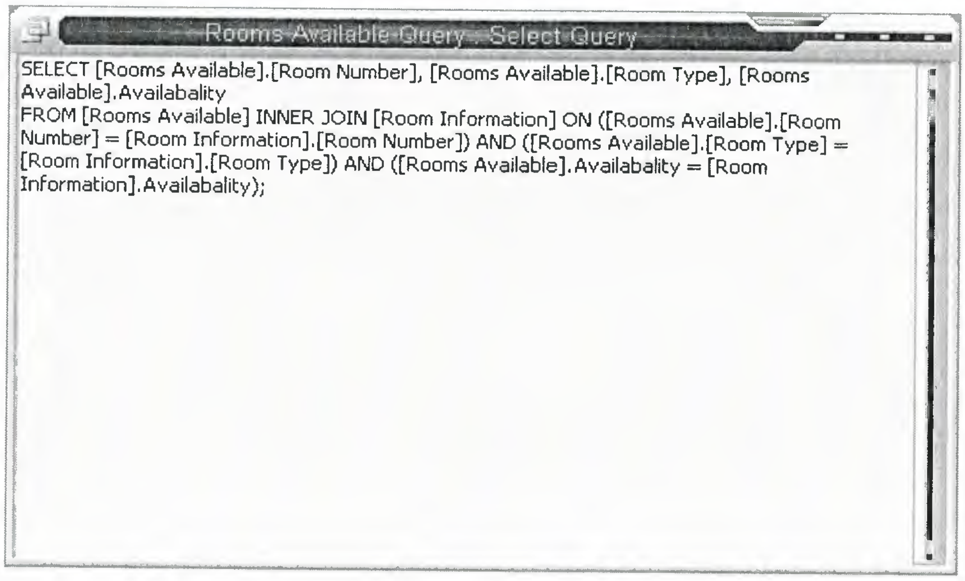


Figure 4.26 Query in Design View

The SQL command has shown in the figure below:



```
SELECT [Rooms Available].[Room Number], [Rooms Available].[Room Type], [Rooms
Available].Availabality
FROM [Rooms Available] INNER JOIN [Room Information] ON ([Rooms Available].[Room
Number] = [Room Information].[Room Number]) AND ([Rooms Available].[Room Type] =
[Room Information].[Room Type]) AND ([Rooms Available].Availabality = [Room
Information].Availabality);
```

Figure 4.27 SQL Command for the Query

4.5 Creating Report for the Hotel

Report is one of the main parts of the database. We can gather all the information of a selected table and can print the data according to the necessity. I created four reports for my project. I used the 'Create report by using Wizard' to create the reports (Figure 4.28).

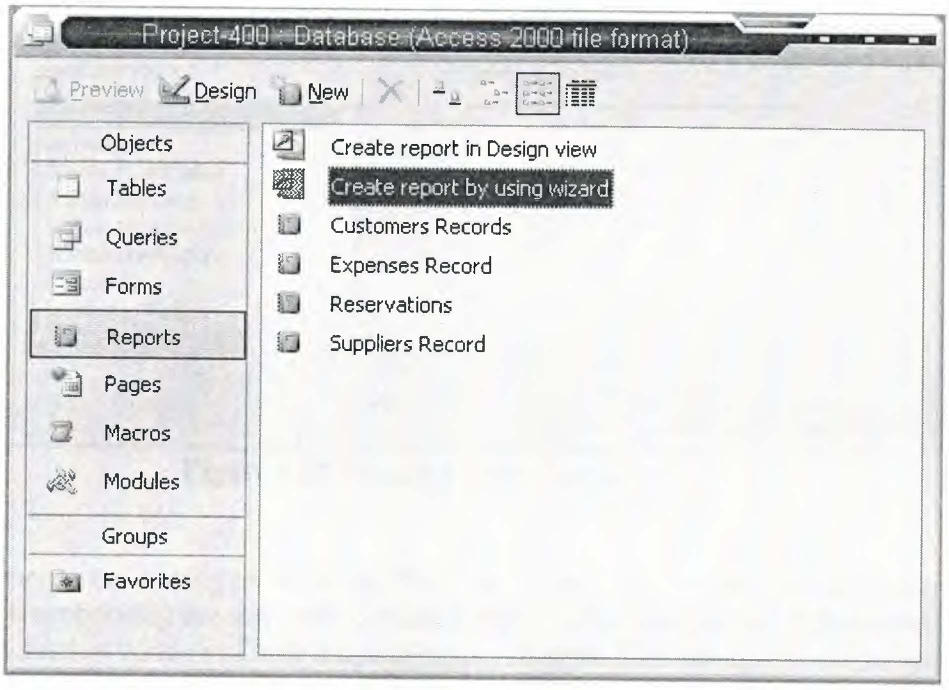


Figure 4.28 Creating report by using wizard

To create the report for 'Accounts' I selected the fields necessary for the report in the 'Selected Fields' by clicking the single arrow (Figure 4.29).

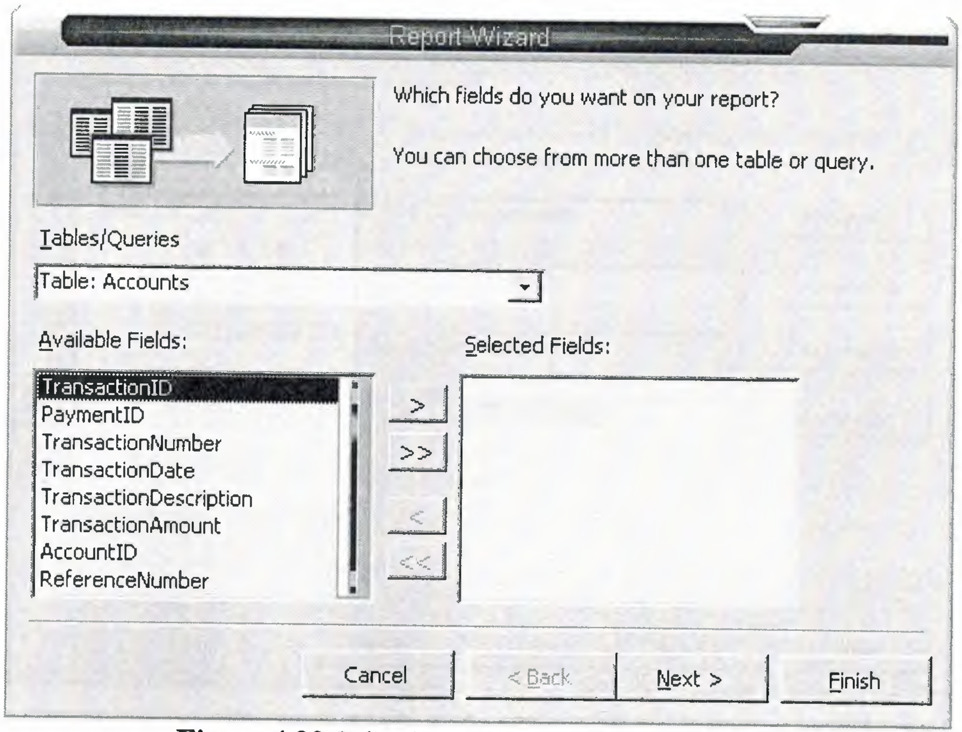
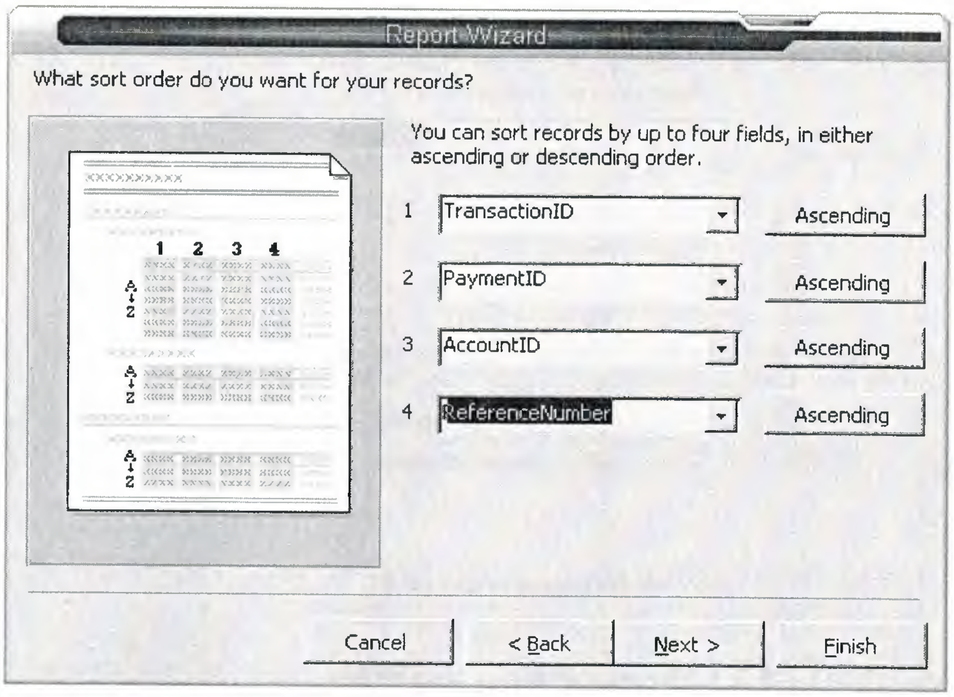


Figure 4.29 Selecting Fields for Report

Then I choose the sorting order for the fields and choose four orders for four fields (Figure 4.30). After choosing the sort order I clicked 'Next'. The window which appeared is the 'Style' selection window. There are many types of style. I choose the 'Bold' style, and then clicked 'Next'. The window appeared is the selecting the 'Title' for the report. I gave the name 'Accounts' and selected 'Preview the report'. The report page appeared.

All the steps I did to create report, I showed them as figure in the next pages as Figure 4.30, Figure 4.31, Figure 4.32 and Figure 4.33.

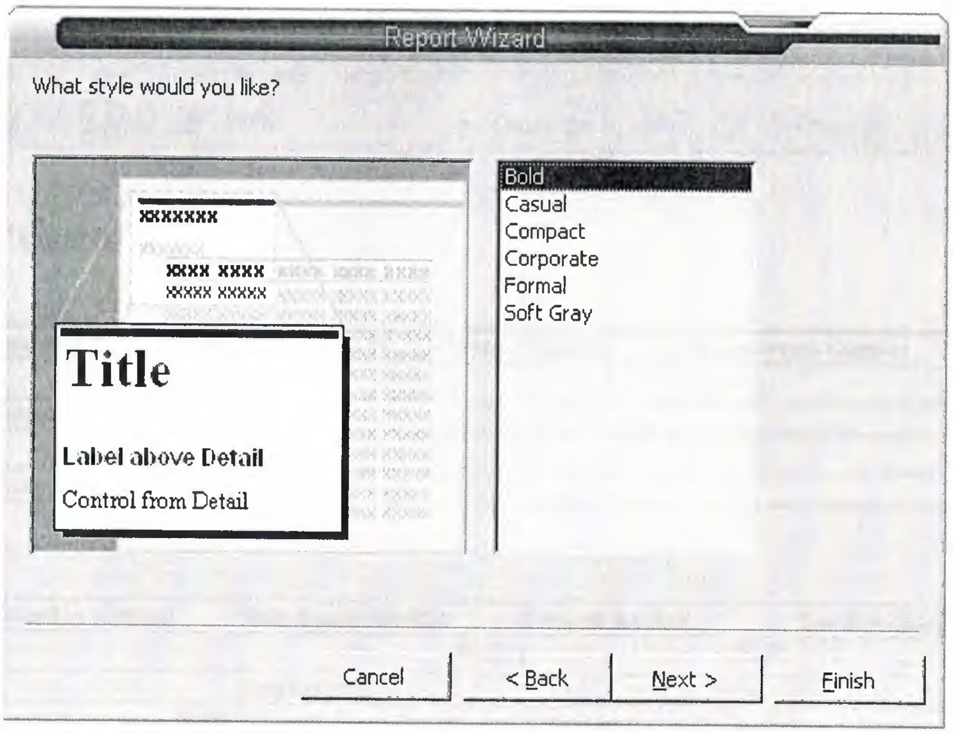


What sort order do you want for your records?

You can sort records by up to four fields, in either ascending or descending order.

1 TransactionID Ascending
2 PaymentID Ascending
3 AccountID Ascending
4 ReferenceNumber Ascending

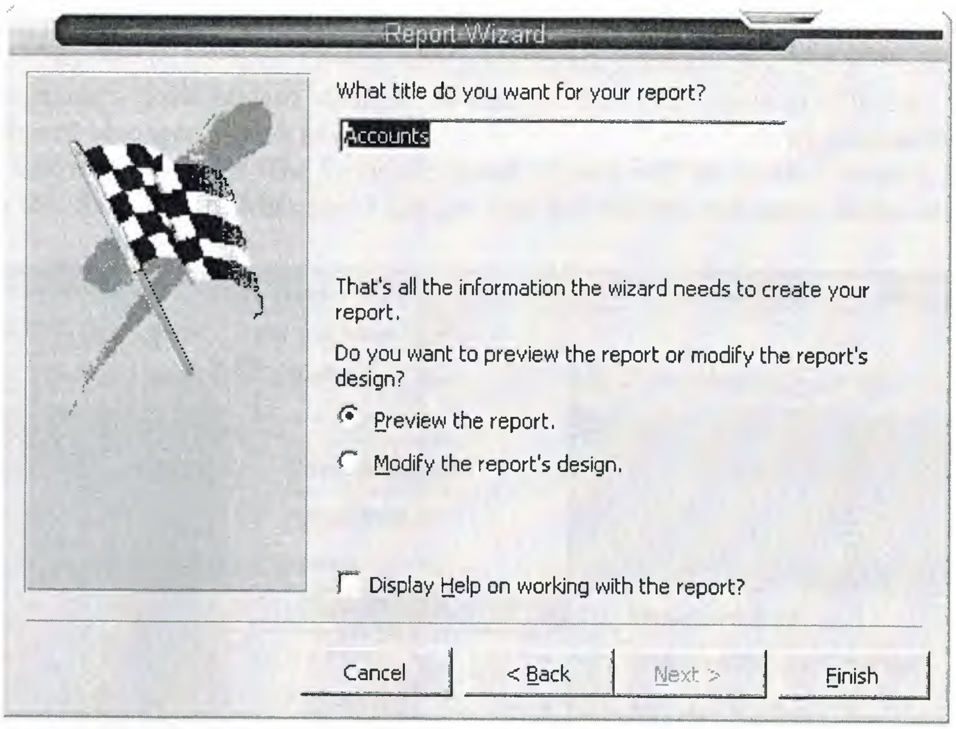
Figure 4.30 Choosing Sort Order



What style would you like?

Bold

Figure 4.31 Choosing the Style



What title do you want for your report?

Accounts

That's all the information the wizard needs to create your report.

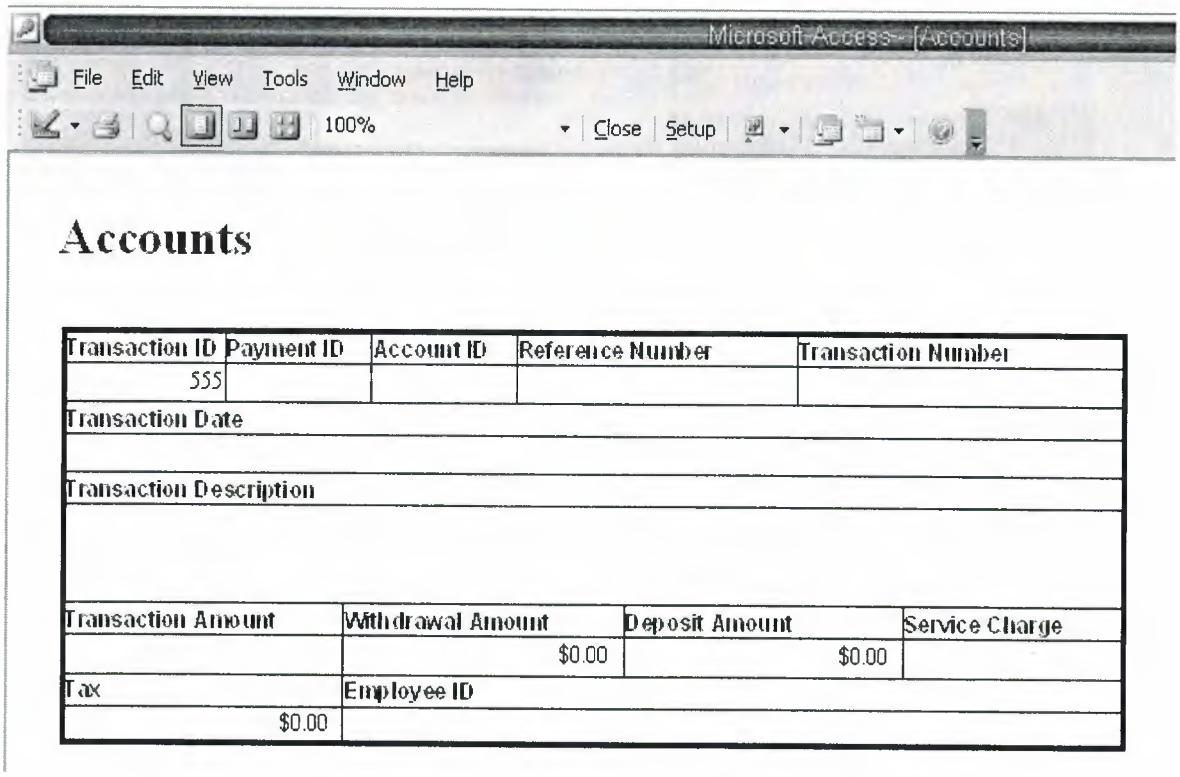
Do you want to preview the report or modify the report's design?

[✓ Preview the report.]

Modify the report's design.

Display Help on working with the report?

Figure 4.32 Giving a Name for the Report


Microsoft Access - [Accounts]
File Edit View Tools Window Help
100%
Close Setup

Accounts

Transaction ID	Payment ID	Account ID	Reference Number	Transaction Number
555				

Transaction Date

Transaction Description

Transaction Amount	Withdrawal Amount	Deposit Amount	Service Charge
	$0.00	$0.00	

Tax	Employee ID
$0.00	

Figure 4.33 Accounts Report

4.6 Applying Switchboard Manager

I need to create a 'Switchboard Manager' to keep all the forms linked in it. In the 'Switchboard Manager' I have to create some buttons of which I have to give names and have to link with the forms. The forms are already linked with the tables I created. So just by creating the 'Switchboard Manager' I can get, add and edit the data saved in the tables.

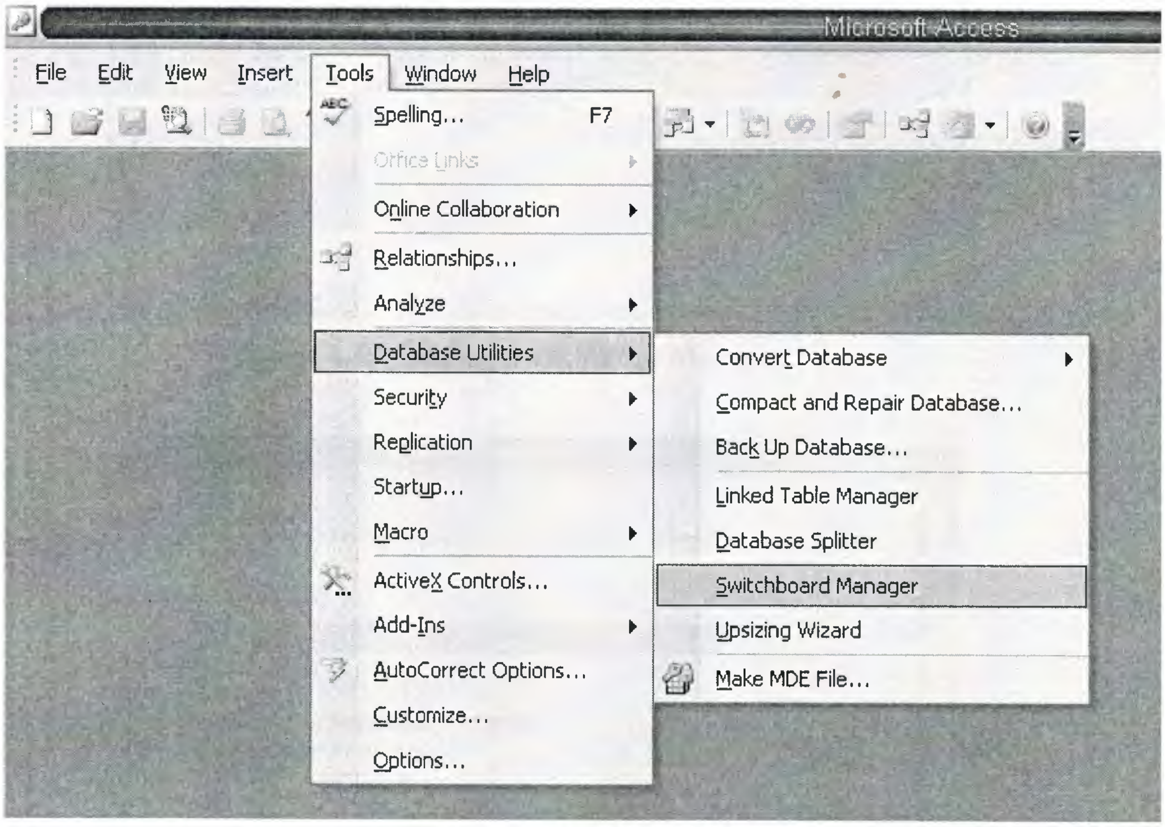
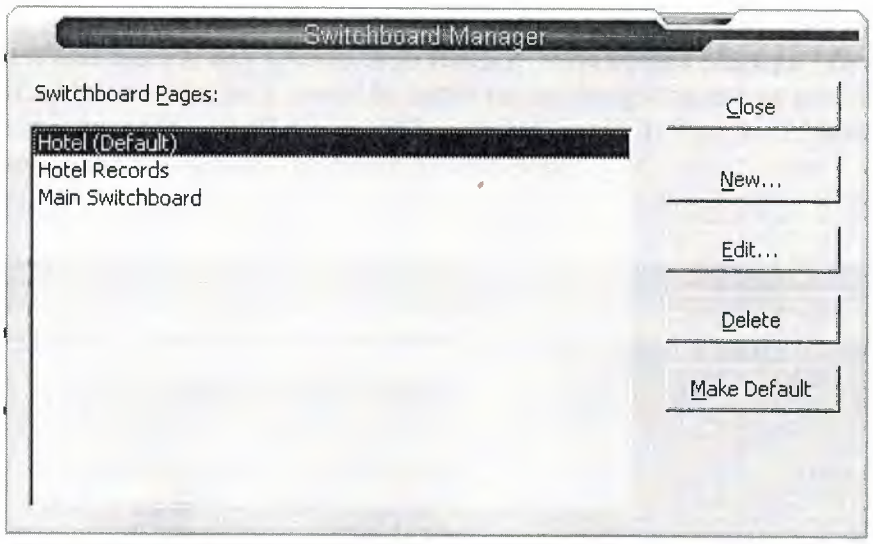


Figure 4.34 Creating the Switchboard Manager

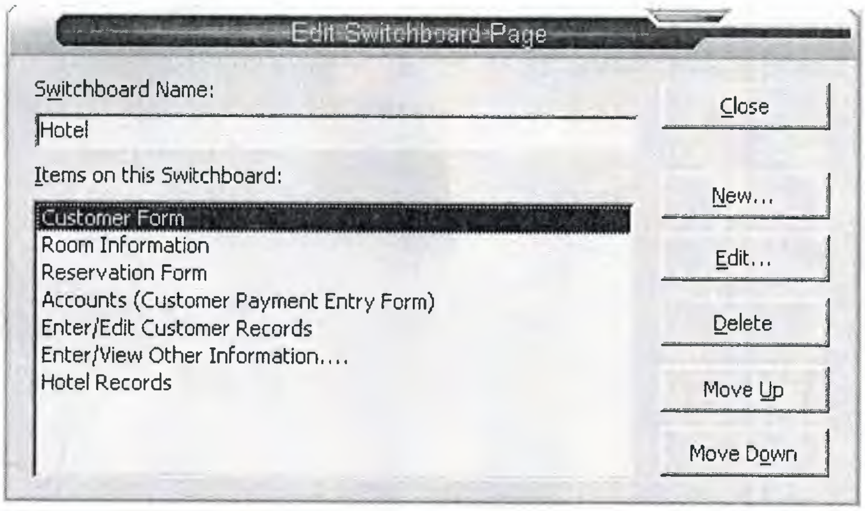
In the database I clicked on 'Tools', highlighted the 'Database Utilities', then clicked the 'Switchboard Manager'. The 'Switchboard Manager' window started where we can create, edit or delete any form or forms. I created a new 'Switchboard Manager' which I named as 'Hotel' and set it as default. Then I created one sub 'Switchboard Manager' form based on the main 'Switchboard Manager' form and I named it as 'Hotel Records'. I double clicked on the 'Hotel' to **edit the switchboard page**. I select the forms which I want to see in this 'Switchboard Manager'. I selected the forms and while double clicking on the forms, the **edit the switchboard items** window opened to edit the forms. I selected the name of the form which I want to see in the switchboard, in the 'Text' option. I gave the command for the form I want to apply on it in the 'Command' option and selected the form I want to use for the command in the 'Form' option. All the steps I did has shown in the figures in the next page as Figure 4.45, Figure 4.46 and Figure 4.47.


Switchboard Manager

Switchboard Pages:

Hotel (Default)
Hotel Records
Main Switchboard

Figure 4.35 Creating Switchboard Manager

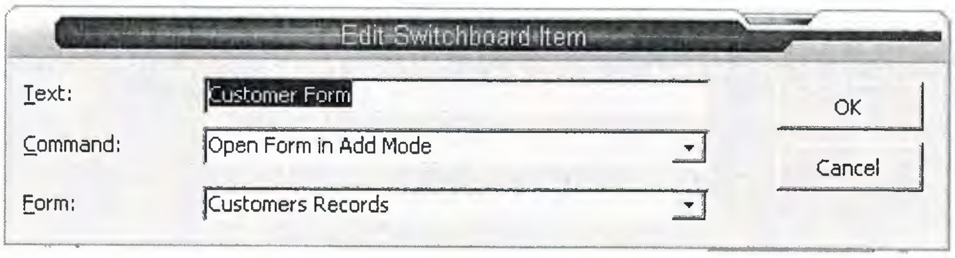

Edit Switchboard Page

Switchboard Name: Hotel

Items on this Switchboard:

Customer Form
Room Information
Reservation Form
Accounts (Customer Payment Entry Form)
Enter/Edit Customer Records
Enter/View Other Information....
Hotel Records

Figure 4.36 Editing Switchboard Page


Edit Switchboard Item

Text: Customer Form
Command: Open Form in Add Mode
Form: Customers Records

Figure 4.37 Editing Switchboard Items

If I want to add any buttons or want to edit the switchboard form I can open the form in the design view and can add any options in it. But the 'Switchboard Manager' form contains a strong Visual Basic code, so it would be better not to change the setting directly. To keep the form safe and the code as well we can add some buttons in it. The 'Switchboard Manager' form in design view looks like the figure given below:

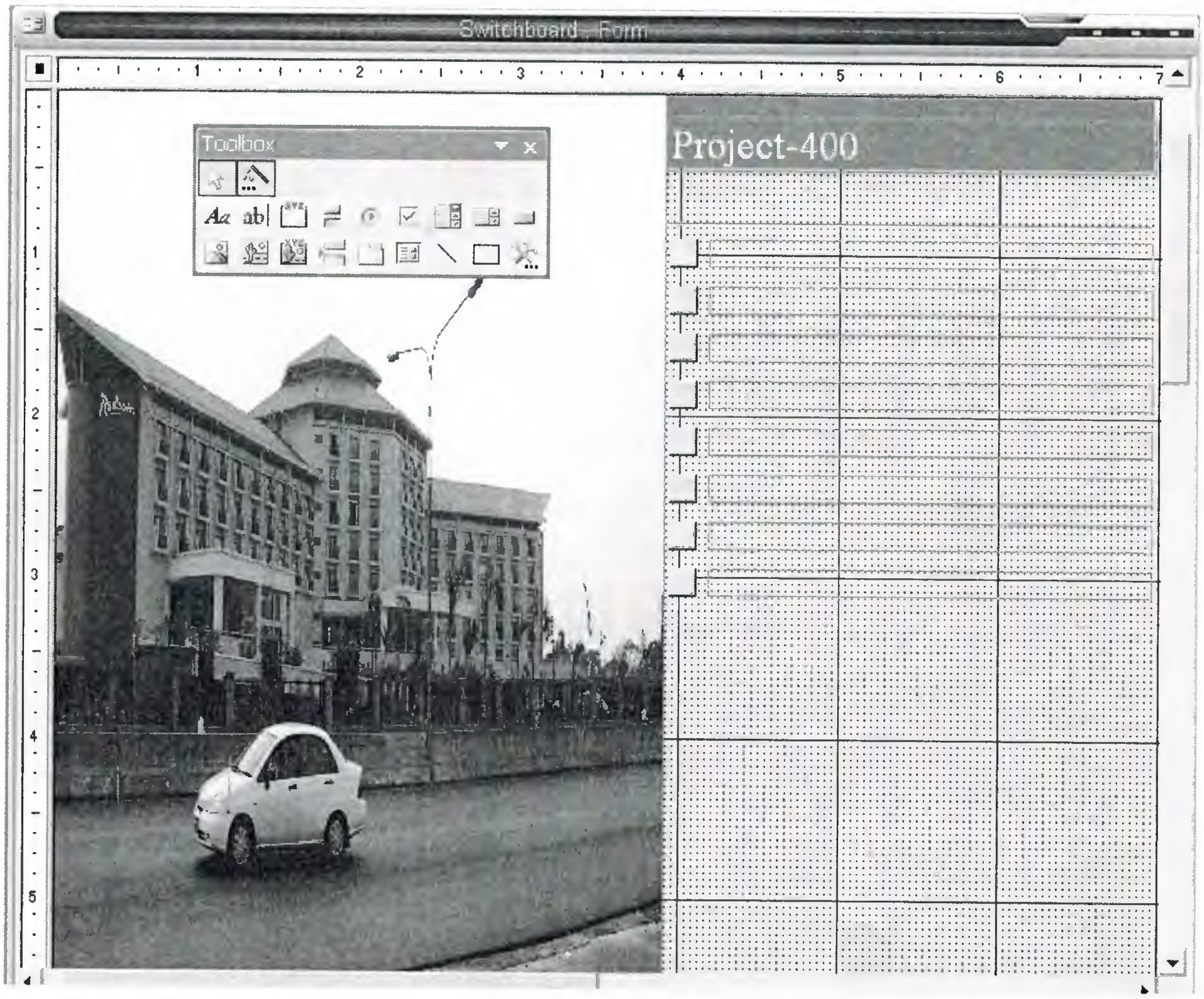


Figure 4.38 Switchboard Form in the Design View

The 'Default' form which I named as 'Hotel' contains seven buttons, by which we can open six different types of forms in different mode and by one we can open (Hotel Records Button) the second page of the 'Switchboard' form which I named as 'Hotel Records'. In the figure below I showed the first Switchboard form and its contents.

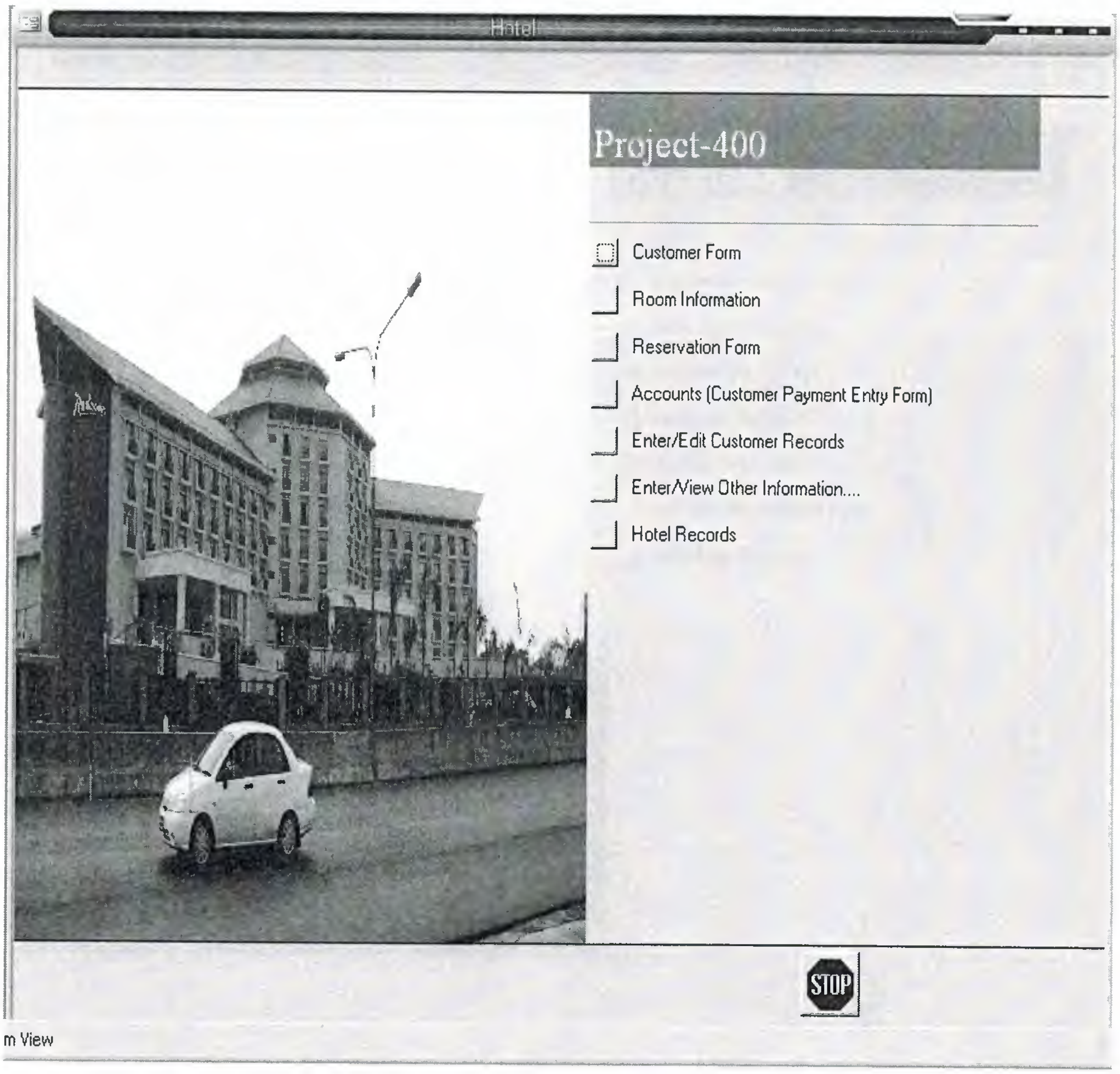

Hotel
Project-400
Customer Form
Room Information
Reservation Form
Accounts (Customer Payment Entry Form)
Enter/Edit Customer Records
Enter/View Other Information....
Hotel Records
STOP
m View

Figure 4.39 Default Switchboard Form Named as 'Hotel'

In the design view I have added one button to stop the whole program. The button's named as 'STOP'.

In the 'Switchboard Form' named as 'Hotel Records' I set seven buttons where six of the buttons will open six different types of form which are very important for the persons working as an administrator, but not necessary to open all the time. One button is to closing the 'Hotel Records' form (Close Hotel Records) which will go back to the 'Hotel' form closing this 'Hotel Records' form. The figure of the 'Hotel Records' form has shown below:

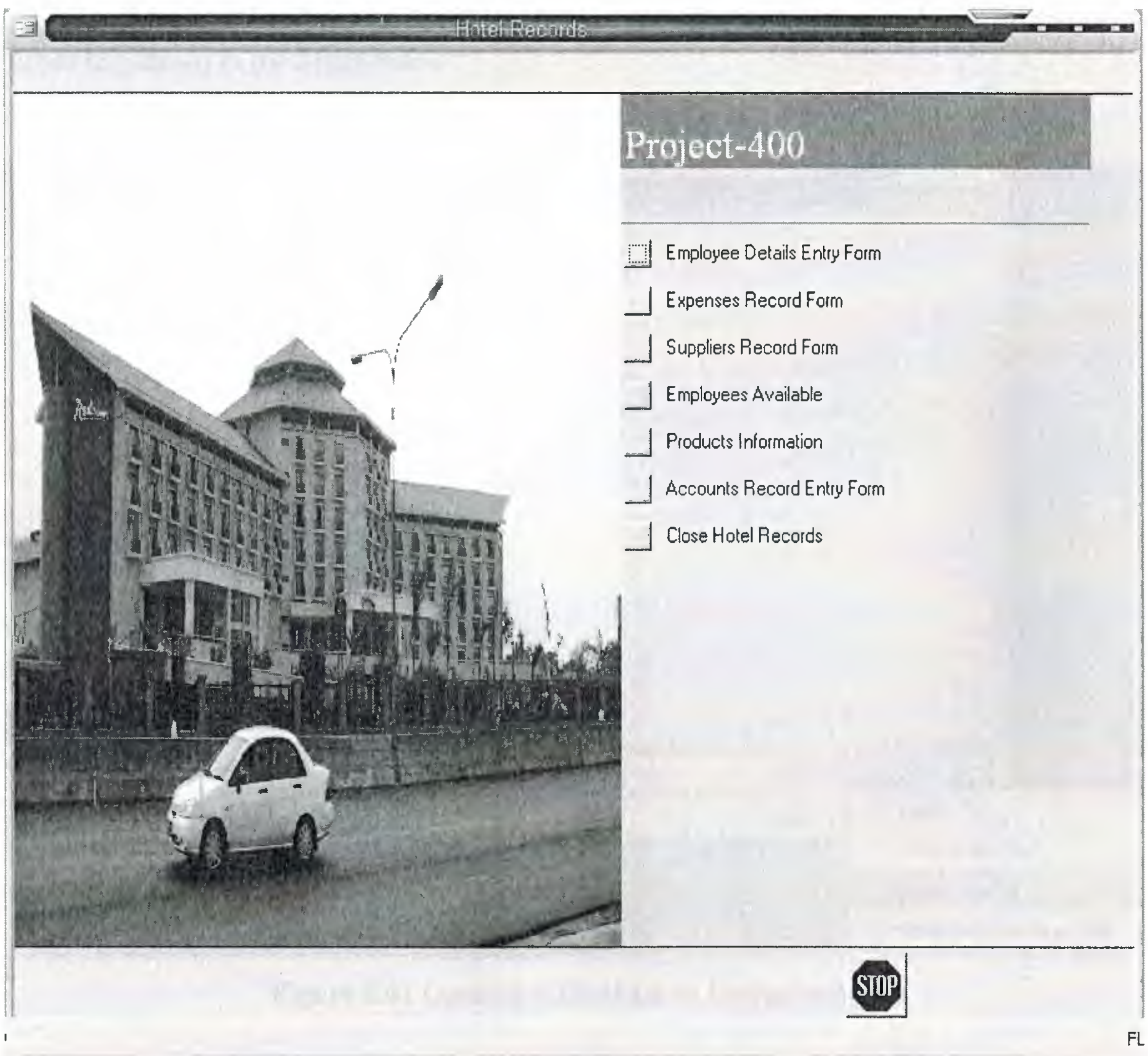


Figure 4.40 Second Switchboard Form Named as 'Hotel Records'

4.7 Applying security password to database

I need to put a password for my database so that no one can change the data or settings of the database. To put a password I need to open the database I created for the 'Hotel' which is named as 'Project-400' in the exclusive mode. I selected the name of the database in the open window. Then in the open button, which is on the lower right corner I selected the drop down box and selected the 'Open Exclusive'. The steps to open database in the open exclusive mode has shown in the figure below:

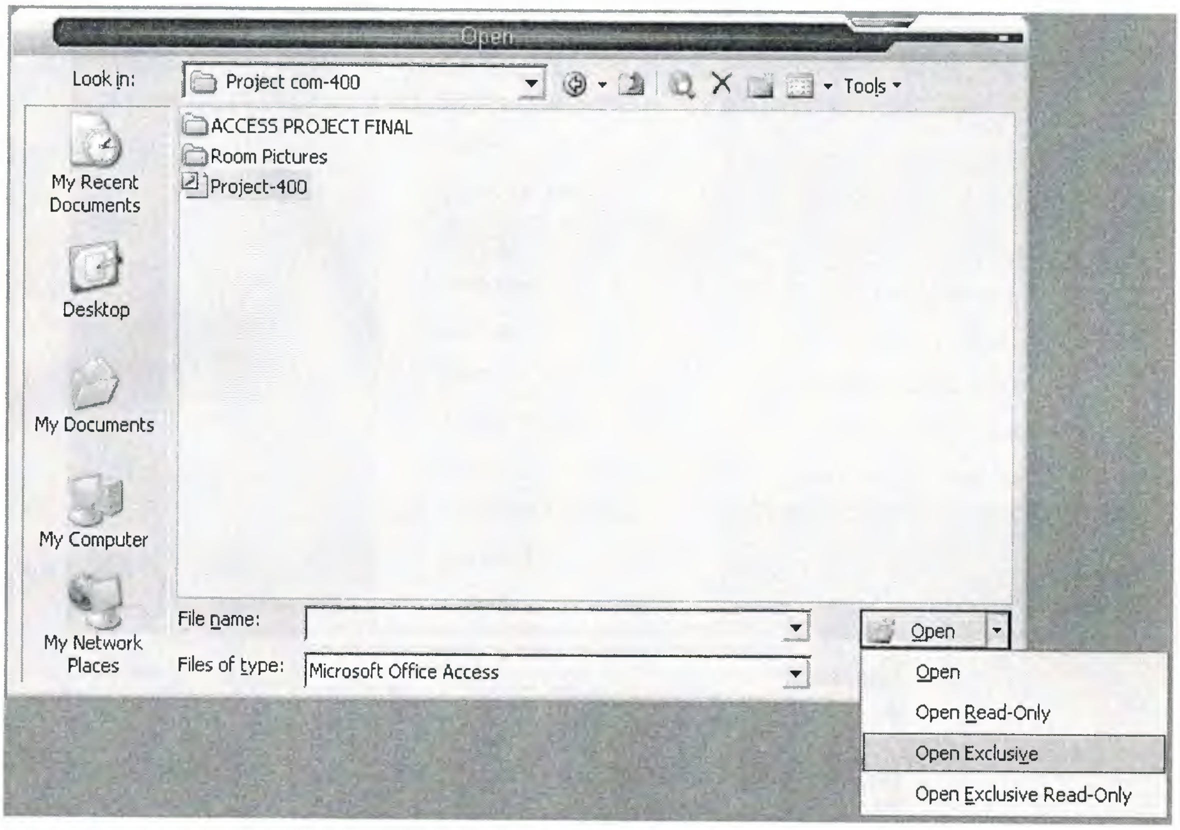


Figure 4.41 Opening a Database in Exclusive Mode

The last step to create a password for the database I created, is to open the database and selecting the 'tools' button. In the 'Tools' I need to highlight the security and then have to select the 'Set Database Password…' option. The password giving window opens where I need to give the password to set and then have to verify it. The figure to set database password has given below:

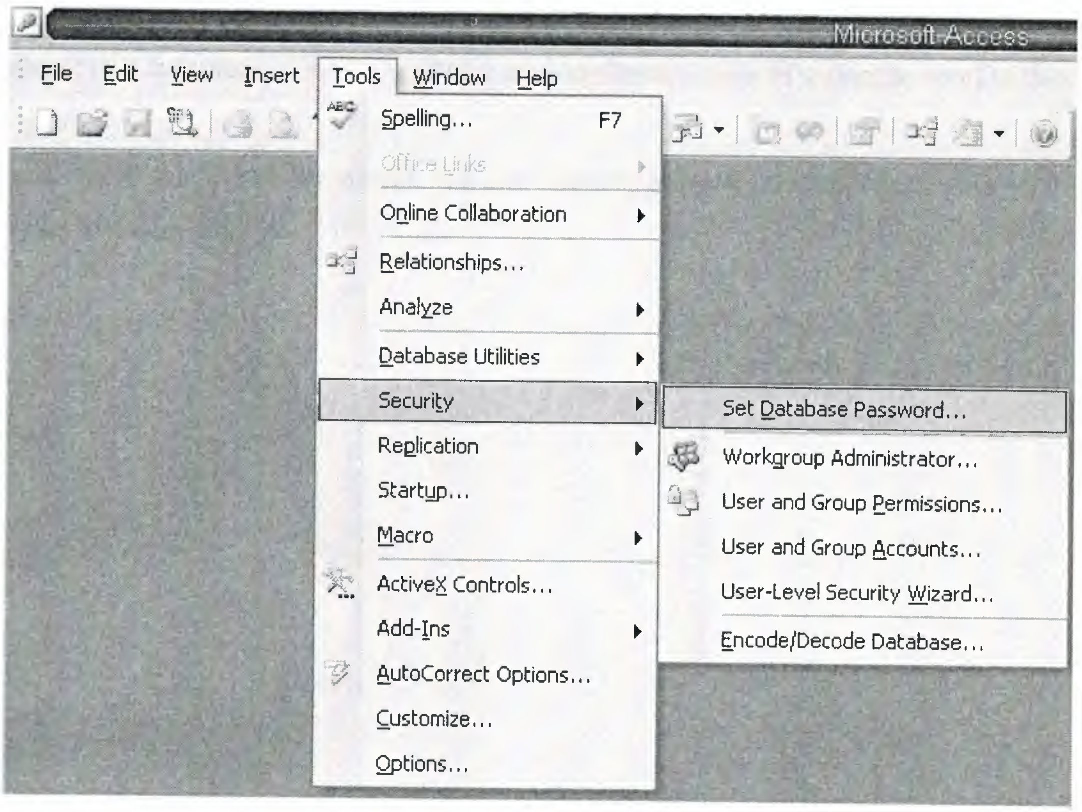


Figure 4.42 Steps to Set Database Password

CONCLUSION

Microsoft Access is a powerful database programming tool that could be use in many ways or many purposes to crate customize program to utilize, it can be used to create small database program to large.

The program has been created to be automated a small hotel, It attempted to bring together (for automation) most important sections/departments of a sample hotel in this program.

According to this it's a very simple and user friendly program, it would be beneficiary to anyone to use in practice.

REFERENCES

[1] Evan Callahan. Microsoft Access 2002 Visual Basic: For Applications Step by Step,

[2] Mahbubur Rahman. Microsoft Office 97/2000: MS Access 97/2000, Systech Publication, July 2000.

[3] Bob Villareal. Access 2002 Programming By Example, Que Cooperation Press, July 2002.

[4] www.google.com.

[5] www.microsoft.com.