

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»

Методичні вказівки і завдання
до виконання лабораторних робіт
з курсу «Банківські інформаційні системи»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

Методичні вказівки і завдання
до виконання лабораторних робіт
з курсу «Банківські інформаційні системи»
(для студентів спеціальності 6.03050201 „Економічна кібернетика”)

Укладач:
А. М. Гізатулін, к.е.н., доц.

Розглянуто на засіданні кафедри
прикладної математики і інформатики
Протокол № 7 від 20.12.2010

Затверджено на засіданні
Навчально-видавничої ради ДонНТУ
Протокол № __ від __.__.2011

УДК 004.052:336.71

Методичні вказівки і завдання до виконання лабораторних робіт з курсу «Банківські інформаційні системи» / для студентів спеціальності 6.03050201 Економічна кібернетика / Укладач доц. Гізатулін А.М. – Донецьк: ДонНТУ, 2011. – 72 с.

Методичні вказівки і завдання до виконання лабораторних робіт з курсу «Банківські інформаційні системи» підготовлені на основі типової програми курсу і направлені на вивчення методології, методики та інструментарію побудови банківських інформаційних систем, їх аналіз та використання. Метою лабораторного практикуму є формування системи практичних знань у галузі дослідження та проектування інформаційних систем у банківській сфері.

Запропоновані лабораторні роботи виконуються за допомогою сучасної систем управління базами даних Microsoft Access 2007 і містять докладні приклади розв'язання поставлених задач.

Укладач:

А.М. Гізатулін, к.е.н., доц.

TABLE OF CONTENTS

Lab 1. Creating Tables	5
Lab 2. Creating Queries	22
Lab 3. Creating Multiple Table Queries	29
Lab 4. Creating Table Form	33
Lab 5. Creating Table Report	39
Lab 6. Creating Macros	47
Lab 7. Creating Conditional Expressions and Assigning a Macro	53
Lab 8. Creating Module	61
Task variants	67
Appendix A Macro action reference	68
Appendix B Data model for combined investment and retail banking operational data	71
Literature	72

LAB 1 CREATING TABLES

Lab objective: to study process of creating tables

The first databases implemented during the 1960s and 1970s were based upon either flat data files or the hierarchical or networked data models. These methods of storing data were relatively inflexible due to their rigid structure and heavy reliance on applications programs to perform even the most routine processing.

In the late 1970s, the *relational database model* which originated in the academic research community became available in commercial implementations such as IBM DB2 and Oracle. The relational data model specifies data stored in *relations* that have some *relationships* among them (hence the name *relational*).

In relational databases such as Sybase, MySQL, Oracle, IBM DB2, MS SQL Server and MS Access, data is stored in *tables* made up of one or more *columns* (Access calls a column a *field*). The data stored in each column must be of a single *data type* such as Character (sometimes called a "string"), Number or Date. A collection of values from each column of a table is called a *record* or a *row* in the table.

Different tables can have the same column in common. This feature is used to explicitly specify a relationship between two tables. Values appearing in column A in one table are shared with another table.

Below are two examples of tables in a relational database for a local bank:

Customer Table

CustomerID	Name	Address	City	State	Zip
<i>Number</i>	<i>Character</i>	<i>Character</i>	<i>Character</i>	<i>Character</i>	<i>Character</i>
1001	Mr. Smith	123 Lexington	Smithville	KY	91232
1002	Mrs. Jones	12 Davis Ave.	Smithville	KY	91232
1003	Mr. Axe	443 Grinder Ln.	Broadville	GA	81992
1004	Mr. & Mrs. Builder	661 Parker Rd.	Streetville	GA	81990

The Customer table has 6 columns (CustomerID, Name, Address, City, State and Zip) and 4 rows (or records) of data. The Accounts table has 5 columns (CustomerID, AccountNumber, AccountType, DateOpened and Balance) with 7 rows of data.

Each of the columns conforms to one of three basic *data types*: Character, Number or Date. The data type for a column indicates the type of data values that may be stored in that column.

Number - may only store numbers, possibly with a decimal point.

Character - may store numbers, letters and punctuation. Access calls this data type **Text**.

Date - may only store date and time data.

In some database implementations other data types exist such as Images (for pictures or other data). However, the above three data types are most commonly used.

Accounts Table

CustomerID	AccountNumber	AccountType	DateOpened	Balance
<i>Number</i>	<i>Number</i>	<i>Character</i>	<i>Date</i>	<i>Number</i>

1001	9987	Checking	10/12/1989	4000.00
1001	9980	Savings	10/12/1989	2000.00
1002	8811	Savings	01/05/1992	1000.00
1003	4422	Checking	12/01/1994	6000.00
1003	4433	Savings	12/01/1994	9000.00
1004	3322	Savings	08/22/1994	500.00
1004	1122	Checking	11/13/1988	800.00

Notice that the two tables share the column `CustomerID` and that the values of the `CustomerID` column in the `Customer` table are the same the values in the `CustomerID` column in the `Accounts` table. This *relationship* allows us to specify that the Customer **Mr. Axe** has both a Checking and a Savings account that were both opened on the same day: December 1, 1994.

Another name given to such a relationship is *Master/Detail*. In a master/detail relationship, a single master record (such as Customer 1003, Mr. Axe) can have many details records (the two accounts) associated with it.

In a Master/Detail relationship, it is possible for a Master record to exist without any Details. However, it is impossible to have a Detail record without a matching Master record. For example, a Customer may not necessarily have any account information at all. However, any account information *must* be associated with a single Customer.

Each table also must have a special column called the **Key** that is used to uniquely identify rows or records in the table. Values in a key column (or columns) may never be duplicated. In the above tables, the `CustomerID` is the key for the `Customer` table while the `AccountNumber` is the key for the `Accounts` table.

A Business Example

In this section, we will outline a business example that will be used as a basis for the examples throughout the tutorial. In organizations, the job of analyzing the business and determining the appropriate database structure (tables and columns) is typically carried out by *Systems Analysts*. A Systems Analyst will gather information about how the business operates and will form a *model* of the data storage requirements. From this model, a database programmer will create the database tables and then work with the application developers to develop the rest of the database application.

For this tutorial, we will consider a simple banking business. The bank has many customers who open and maintain one or more accounts. For each Customer, we keep a record of their name and address. We also assign them a unique `CustomerID`. We assign this unique identifier both for convenience and for accuracy. It is much easier to identify a single customer using their `CustomerID` rather than by looking up their full name and address. In addition, it is possible for the bank to have two customers with the same name (e.g., Bill Smith). In such cases, the unique `CustomerID` can always be used to tell them apart.

In a similar fashion, all accounts are assigned a unique account number. An account can be either a checking account or a savings account. Savings accounts earn interest but the only transactions allowed are deposits and withdrawals. Checking accounts do not earn interest. We maintain the date that the account was opened. This helps us track our customers and can be useful for marketing purposes. Finally, we maintain the current balance of an account.

In the previous section, we gave the structure and some sample data for the `Customer` table and the `Accounts` table. These will be used to support the data storage part of our Banking application.

In any database application, each of the tables requires a means to get data into them and retrieve the data at a later time. The primary way to get data into tables is to use data entry forms. The primary ways to get data back out of tables or to display data in tables are to use queries or reports.

For this tutorial, we will create a data entry form for each table, a query for each table and a report for each table.

In the following sections, we will first introduce how to start Access and how to create a new database.

Starting Microsoft Access

As with most Windows programs, Access can be executed by navigating the Start menu in the lower left-hand corner of the Windows Desktop.

To start Access, click on the Start button, then the Programs menu, then move to the MS Office menu and finally click on the Microsoft Access menu item. The MS Office Professional menu is shown below.



Note that this arrangement of menus may vary depending on how MS Office was installed on the PC you are using.

Once Access is running, an initial screen will be displayed:



From this initial screen, the user can create a new database (either blank or with some tables created with the database wizard), or open up an existing database.

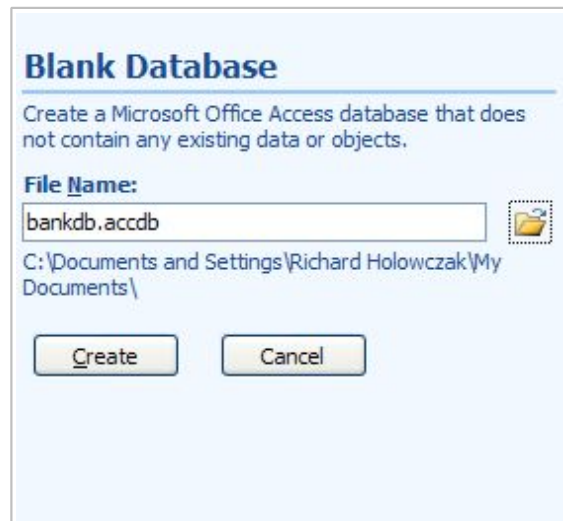
In general, the first time one begins a project, a new, blank database should be created. After that point, use the *Open existing database* option to re-open the database created previously.

Warning - If you have previously created a database, and then create it again using the same name, you will overwrite any work you have done.

For the purposes of this tutorial, if you are going through these steps for the first time, choose the option to create a new, blank database as shown in the above figure.

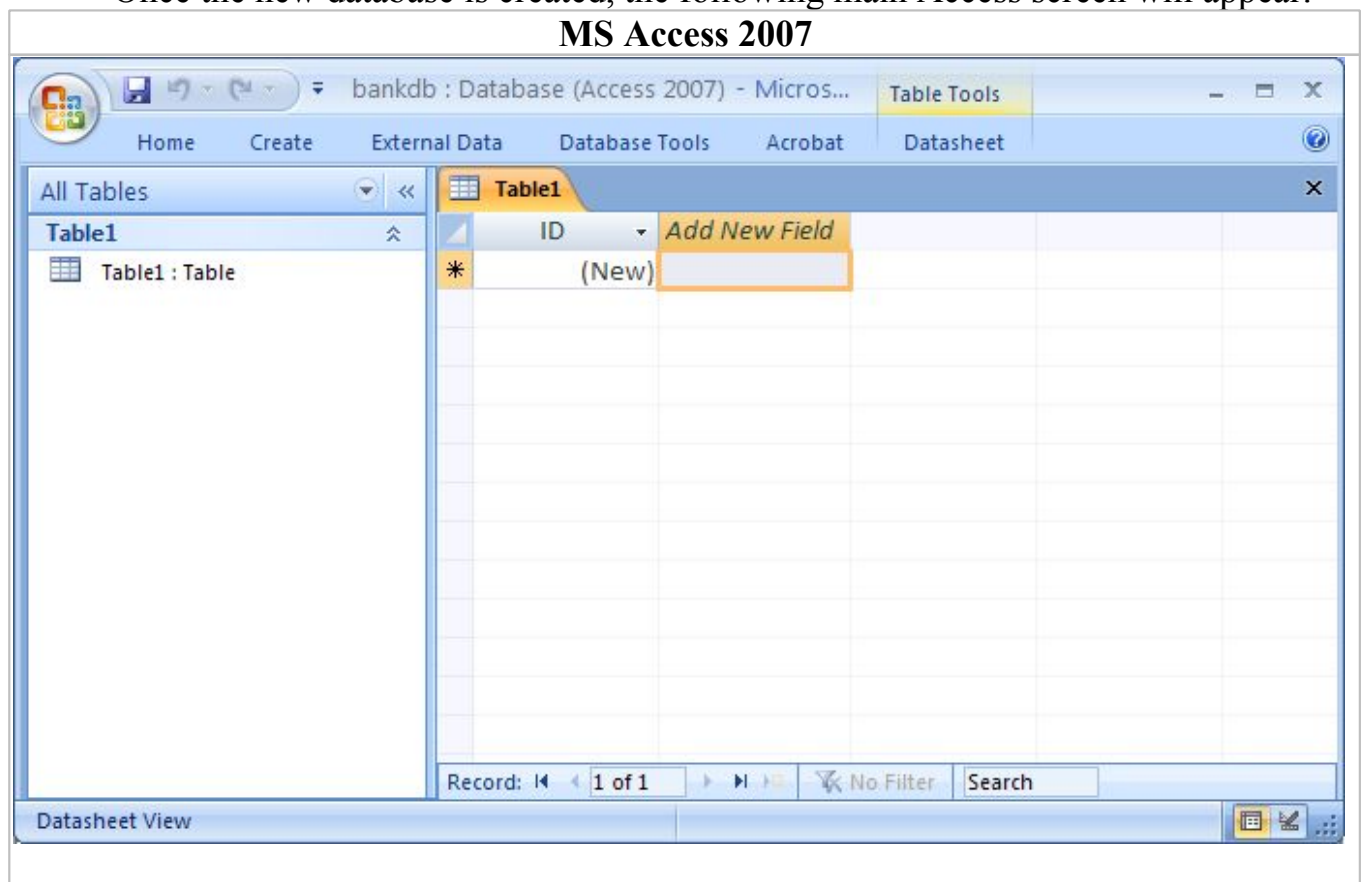
In Access 2007, click on the round Office button in the upper left corner and choose New from the drop down menu. Fill in *File Name* as `bankdb.accdb` and click on the `Create` button to create the database as in the figure below.

It is advisable to keep the name of the database (`bankdb` in the above example) relatively short and do not use spaces or other punctuation in the name of the database. Also, the name of the database should reflect the database's contents.



New Database screen for Access 2007
 In the above file name, *bankdb* is the name chosen for this particular database and *.accdb* is the file name extension given for *Microsoft DataBase 2007* files.

Once the new database is created, the following main Access screen will appear:



The screen layout for MS Access 2007 is significantly different from past versions. Most of the tabs along the top of the screen have been rearranged. In addition, the default main screen after creating a new database automatically switches to the Design view to create a new table.

The following tabs will appear at the top of the screen:

- **Home tab** - Controls for changing fonts, performing queries, copy/paste/cut data, etc.
- **Create tab** - Controls for creating tables, forms, reports, etc.
- **External Data tab** - Controls for loading data from other data sources into MS

Access.

- **Database Tools tab** - Controls for managing databases (security, switchboard, etc.)
- **Design tab** - This will appear when designing a new table, form, report, etc.

Review of Starting Microsoft Access

To start Microsoft Access:

1. Use the `Start` button on the task bar to open: `Programs -> MS Office -> Microsoft Access`
2. To create a new database, choose **Blank Database** and specify a new file name for the database. Be sure to use a descriptive name for the new database. Click on the `OK` button to create the new database.
3. To open an existing database, choose **Open an Existing Database**, highlight *More Files...* and click on the `OK` button. Then navigate to the drive, highlight the existing database file on the floppy disk and click the `OK` button again to open the database.

To exit Access, pull down the `File` menu (or `Office` menu) and select the `Exit` menu item.

Creating and Viewing Tables

Tables are the main units of data storage in Access. Recall that a table is made up of one or more *columns* (or *fields*) and that a given column may appear in more than one table in order to indicate a relationship between the tables.

From the business example discussed earlier, we concluded that two tables would be sufficient to store the data about **Customers** and their bank **Accounts**. We now give the step-by-step instructions for creating these two tables in Access.

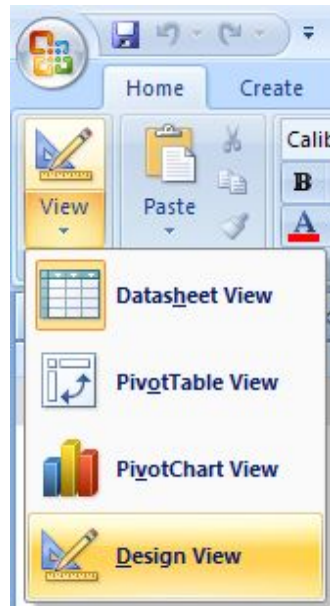
There are a number of ways to create a table in Access. Access provides *wizards* that guide the user through creating a table by suggesting names for tables and columns. The other main way to create a table is by using the *Design View* to manually define the columns (fields) and their data types.

While using the wizards is a fast way to create tables, the user has less control over the column names (fields) and data types. In this tutorial, we will describe the steps to create a table using the *Design View*. Students are encouraged to experiment on their own with using the Create Table wizard.

Creating a Table Using the Design View

To create a table in Access using the Design View, perform the following steps:

1. In Access 2007, the `Create New Table` tab should already be highlighted and a new table named `table1` created. If this is not the case, click on the *Create* tab and click on the *Table* icon. Then pull down the *View* menu and choose *Design View*.



2. The Table Design View will appear. Fill in the **Field Name**, **Data Type** and **Description** for each column/field in the table. The CustomerID field is filled in below:

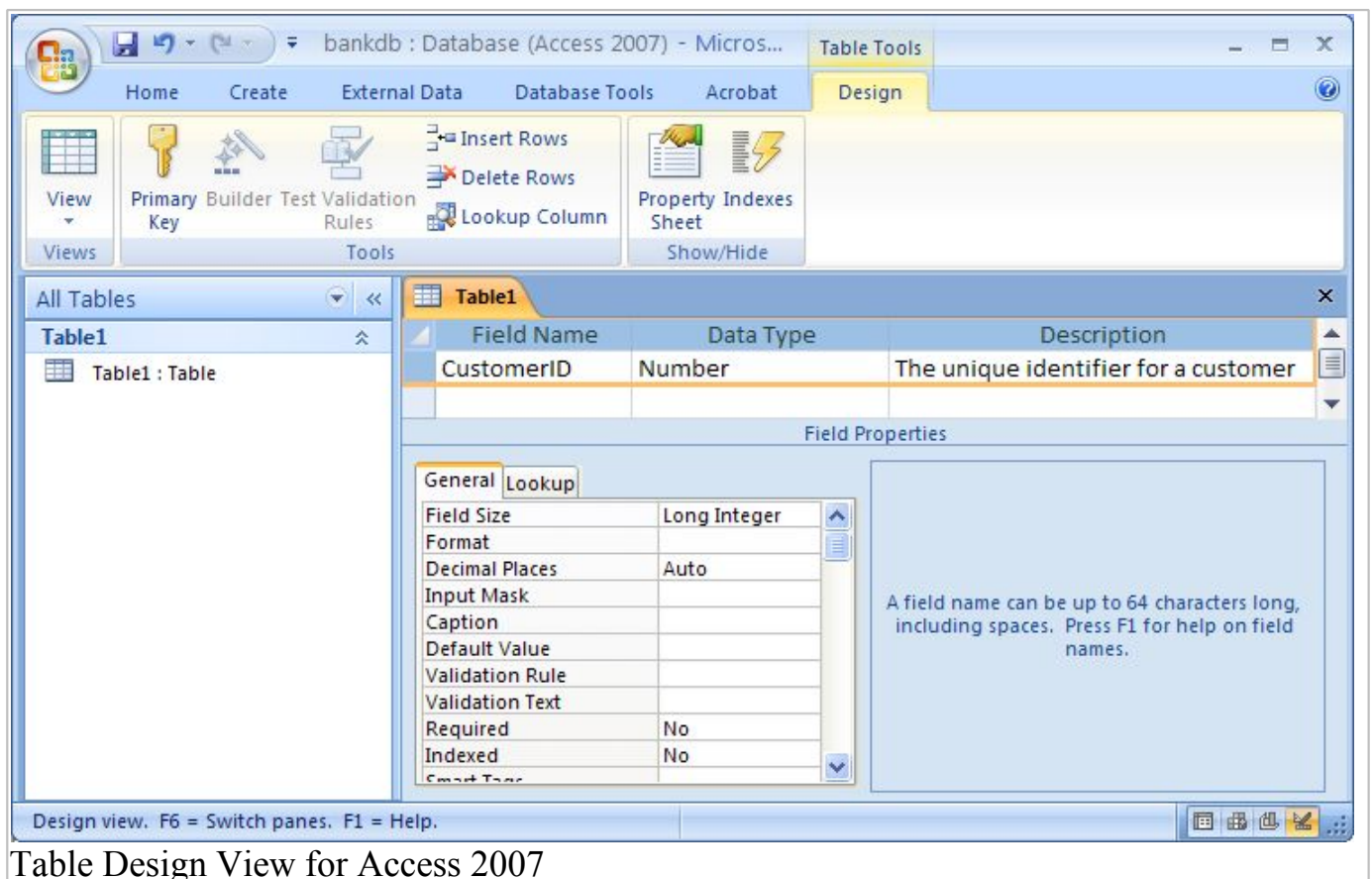


Table Design View for Access 2007

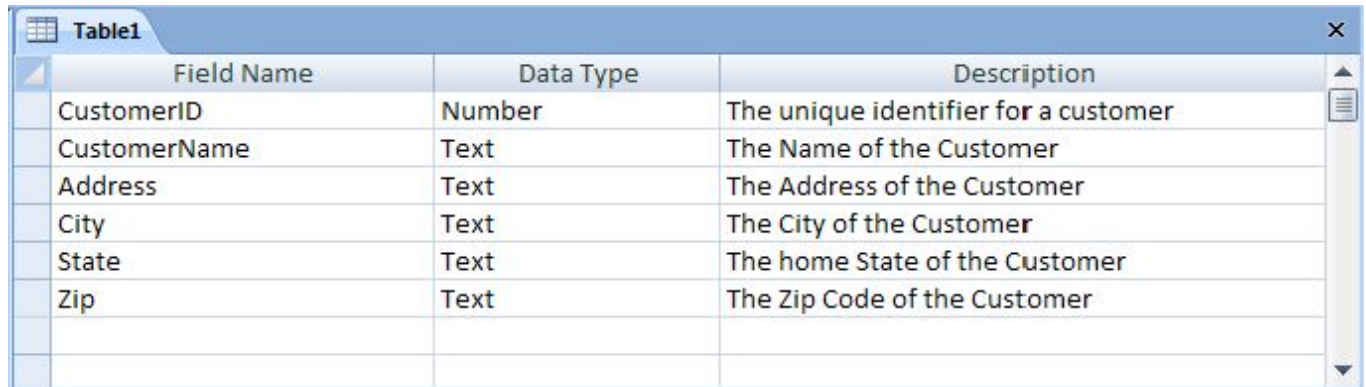
Note that the default name given for the table is Table1. In a later step, we will assign an appropriate name for this table.

3. Fill in the information for the fields as follows:

Field Name	Data Type	Description
CustomerID	Number	The Unique Identifier for a Customer
CustomerName	Text	The Name of the Customer
Address	Text	The Address of the Customer

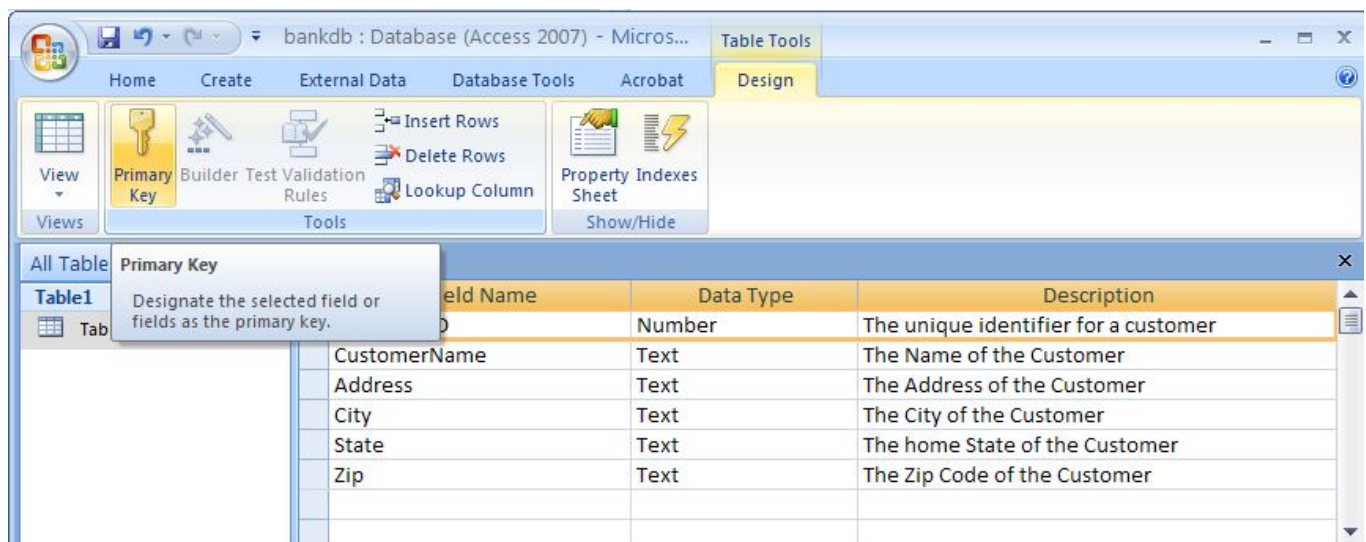
City	Text	The City of the Customer
State	Text	The home State of the Customer
Zip	Text	The Zip Code of the Customer

4. A figure showing the design view with the new table definition filled in is given below:



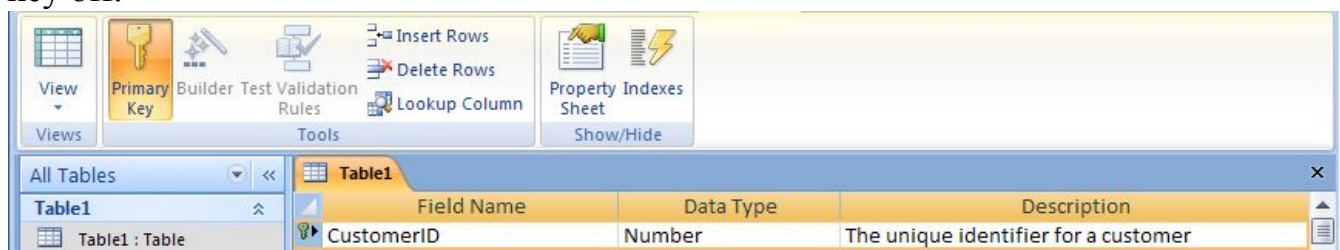
Field Name	Data Type	Description
CustomerID	Number	The unique identifier for a customer
CustomerName	Text	The Name of the Customer
Address	Text	The Address of the Customer
City	Text	The City of the Customer
State	Text	The home State of the Customer
Zip	Text	The Zip Code of the Customer

Now that all of the fields have been defined for the table, a Primary Key should be defined. Recall that the Primary Key will be used to uniquely identify a record in the table (in this case a Customer). Highlight the **CustomerID** field and click on the Primary Key button on the button bar



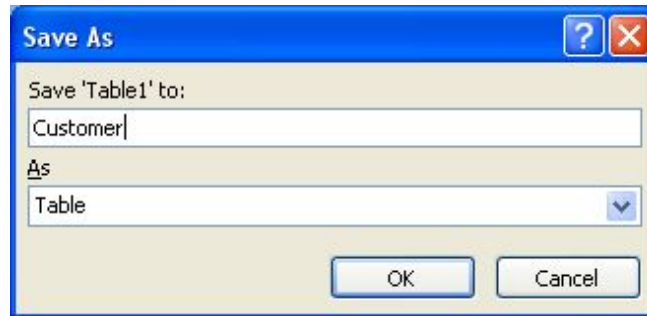
Notice that a small key appears next to the field name on the left side.

Note: To remove a primary key, simply repeat this procedure to toggle the primary key off.



5. As a final step, the table must be saved. Pull down the Office menu and choose the Save As menu item. A dialog box will appear where the name of the new table should be specified. Note that Access gives a default name such as **Table1** or **Table2**. Simply type over this default name with the name of the table. For this example,

name the table: **Customer** Then click on the OK button.



At this point, the new Customer table has been created and saved.

Note about naming fields in MS Access

When defining the fields (columns) for a table, it is important to use field names that give a clear understanding of the data contents of the column. For example, does the field CNO indicate a Customer Number or a Container Number?

Field names in Access can be up to 64 characters long and may contain spaces. **However, the use of spaces in field names and table names is strongly discouraged.** If you wish to make field names easier to read, consider using an underscore character to separate words. However be certain no spaces appear before or after the underscore.

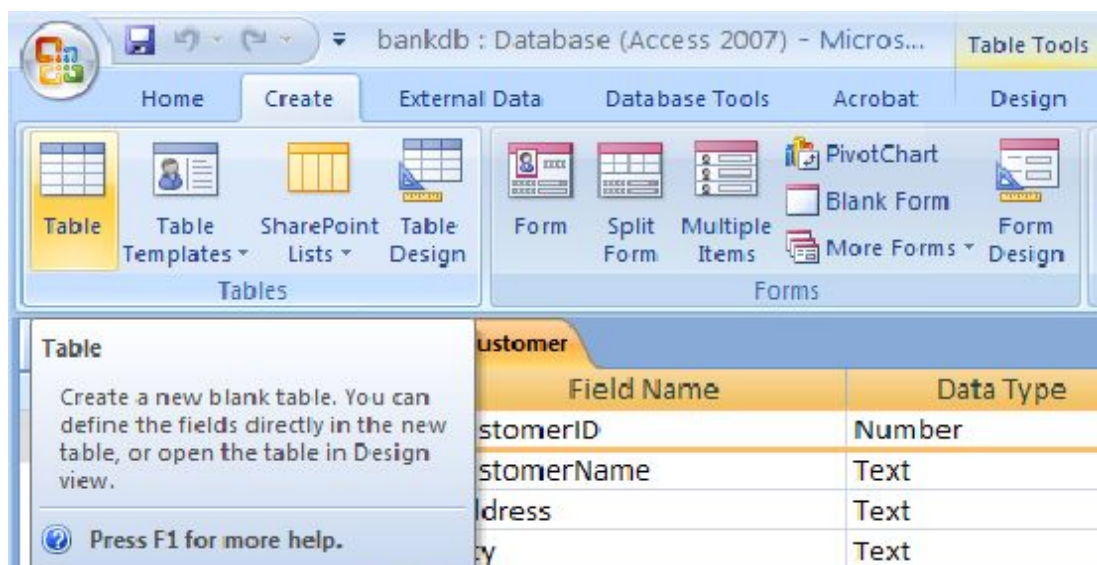
The following table summarizes some different ways to give field names:

Description	Bad	Good
Unique identifier for a customer	CID	CustomerID or Customer_ID
Description for a product	PDESC	ProductDescription
Employee's home telephone number	Employee_home_telephone_number	HomePhone
Bank account number	BA#	AccountNumber

Creating a Table

Create the *Accounts* table by following the same steps used to create the Customer table.

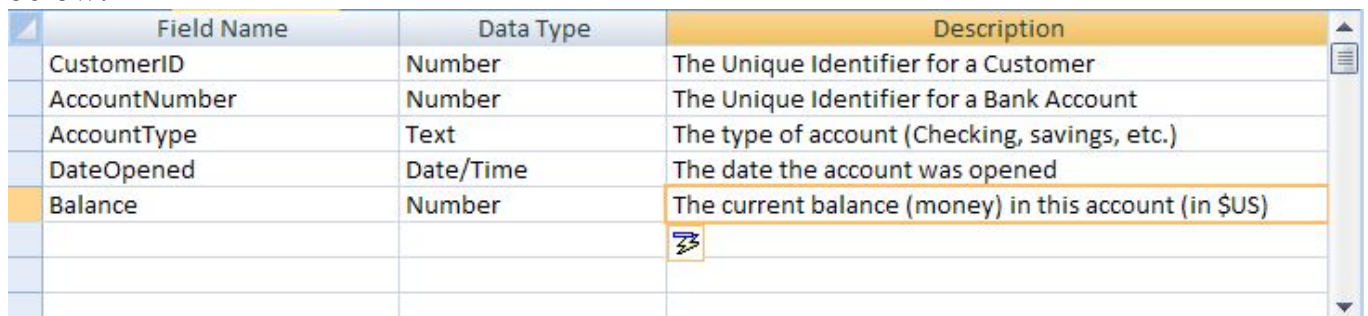
1. Click on the **Create** tab and then click on the *Table* button.



2. Pull down the View menu and choose Design. The Table Design View will appear. Fill in the **Field Name**, **Data Type** and **Description** for each column/field in the Accounts table.

Field Name	Data Type	Description
CustomerID	Number	The Unique Identifier for a Customer
AccountNumber	Number	The Unique Identifier for a Bank Account
AccountType	Text	The type of account (Checking, savings, etc.)
DateOpened	Date	The date the account was opened
Balance	Number	The current balance (money) in this account (in \$US)

3. A figure showing the design view with the new table definition filled in is given below:



Field Name	Data Type	Description
CustomerID	Number	The Unique Identifier for a Customer
AccountNumber	Number	The Unique Identifier for a Bank Account
AccountType	Text	The type of account (Checking, savings, etc.)
DateOpened	Date/Time	The date the account was opened
Balance	Number	The current balance (money) in this account (in \$US)

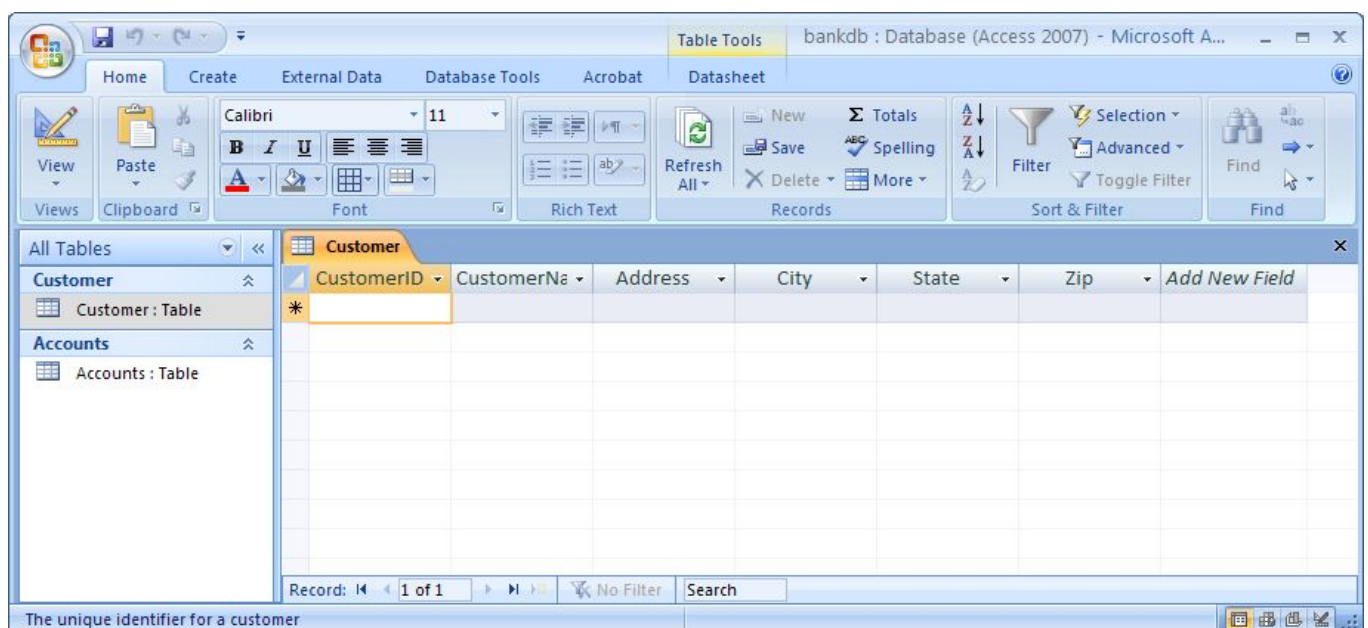
4. Define a Primary Key for the Accounts table. Click on the **AccountNumber** field with the *Right* mouse button and choose Primary Key from the pop-up menu.

5. Save the new Accounts table by pulling down the File menu and choosing the Save menu item. Fill in the name of the table: **Accounts** Then click on the OK button.

Viewing and Adding Data to a Table

Data can be added, deleted or modified in tables using a simple spreadsheet-like display. To bring up this view of a single table's data, highlight the name of the table and then double-click on the name of the table.

In this view of the Customer table, shown in the figure below, the fields (columns) appear across the top of the window and the rows or records appear below. This view is similar to how a spreadsheet would be designed.




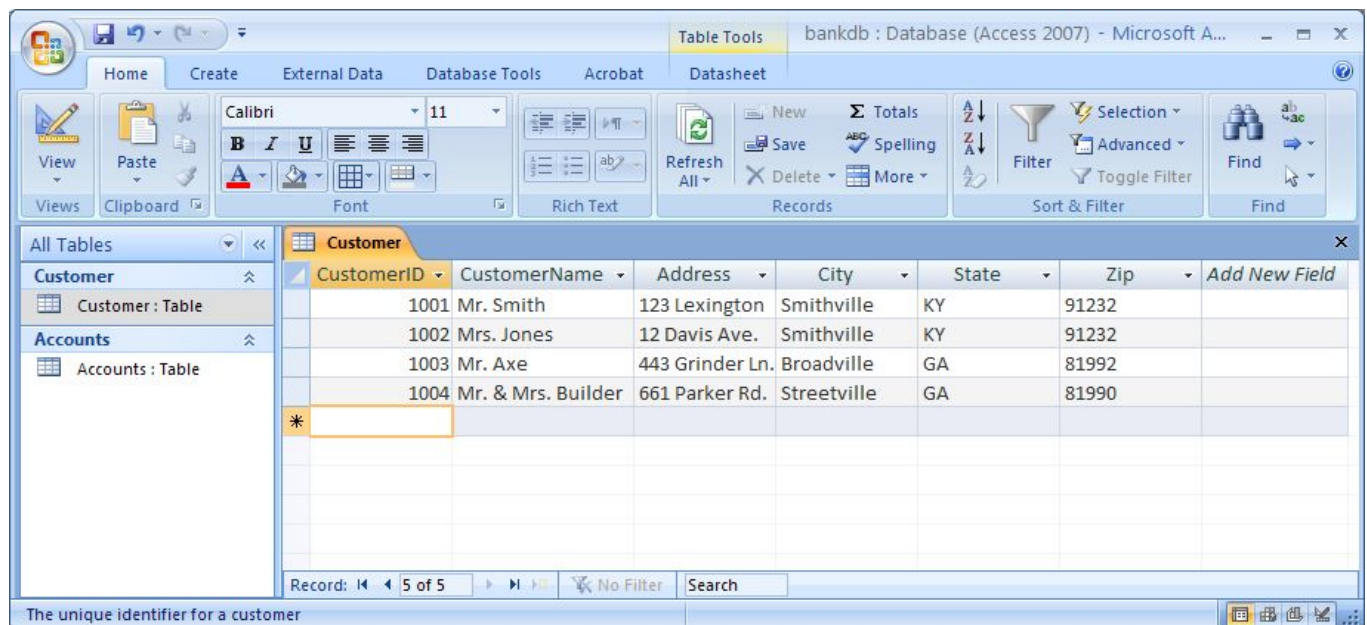
Note at the bottom of the window the number of records is displayed. In this case, since the table was just created, only one blank record appears.

To add data to the table, simply type in values for each of the fields (columns). Press the `Tab` key to move between fields within a record. Use the up and down arrow keys to move between records. Enter the data as given below:

CustomerID	Name	Address	City	State	Zip
1001	Mr. Smith	123 Lexington	Smithville	KY	91232
1002	Mrs. Jones	12 Davis Ave.	Smithville	KY	91232
1003	Mr. Axe	443 Grinder Ln.	Broadville	GA	81992
1004	Mr. & Mrs. Builder	661 Parker Rd.	Streetville	GA	81990

To save the new data, pull down the `Office` menu and choose `Save`.

To navigate to other records in the table, use the navigation bar at the bottom of the screen: Record:  2 of 4



To modify existing data, simply navigate to the record of interest and tab to the appropriate field. Use the arrow keys and the delete or backspace keys to change the existing data.

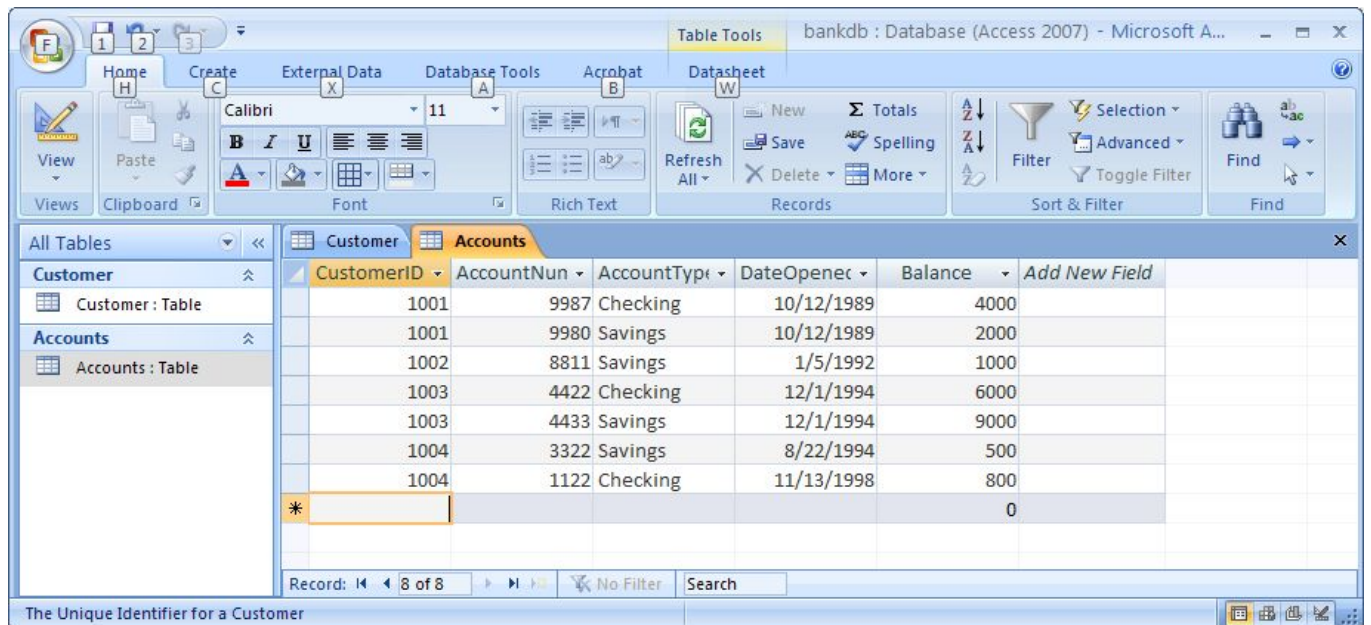
To delete a record, first navigate to the record of interest. Then pull down the `Edit` menu and choose the `Delete` menu item.

To close the table and return to the Access main screen, pull down the `File` menu and choose the `Close` menu item.

Adding Data to a Table

Be sure to enter the data exactly as shown including the capitalization of the data in the `AccountType` field. e.g., type `Savings` instead of `savings` or `SAVINGS`. Note that when entering the dates, type in the full four digits for the year. By default, Access displays all 4 digits of the year (older version of Access only displayed two digits).

Be sure to save the data when you are done. The figure below shows the `Accounts` table and data as it should appear when you are done with this exercise.

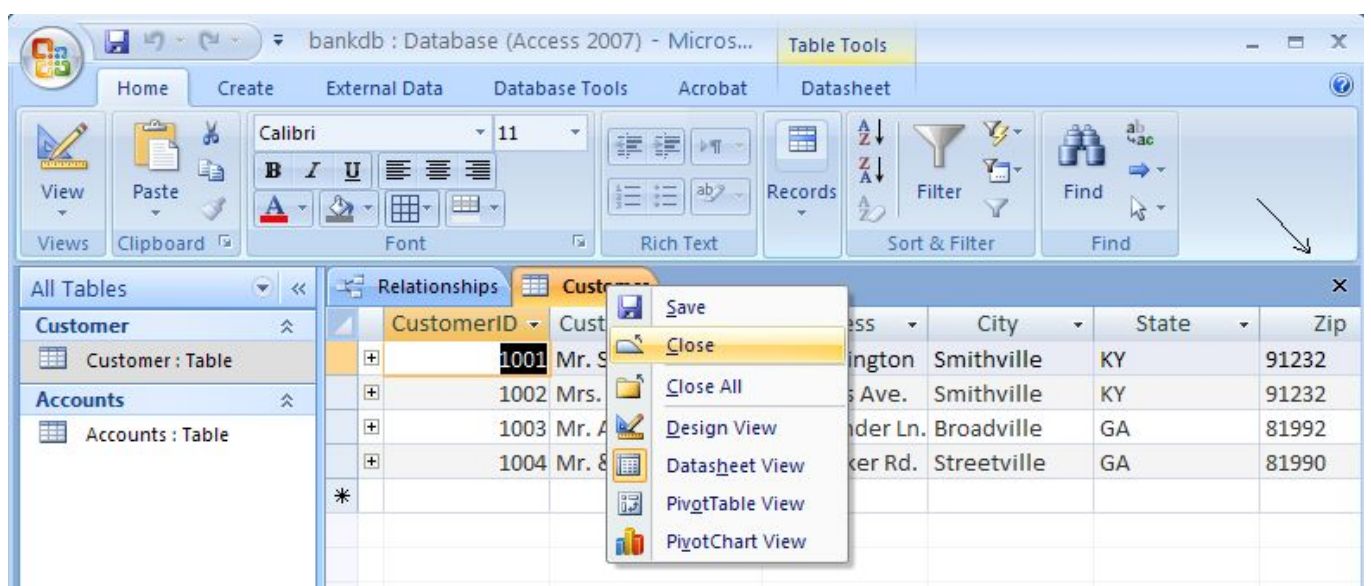


At this point in the tutorial, we have created two tables, Customers and Accounts, and added data to each one. In the subsequent sections, we will cover how to query and report on the data in the tables and how to create a user-friendly data entry form using the Access wizards.

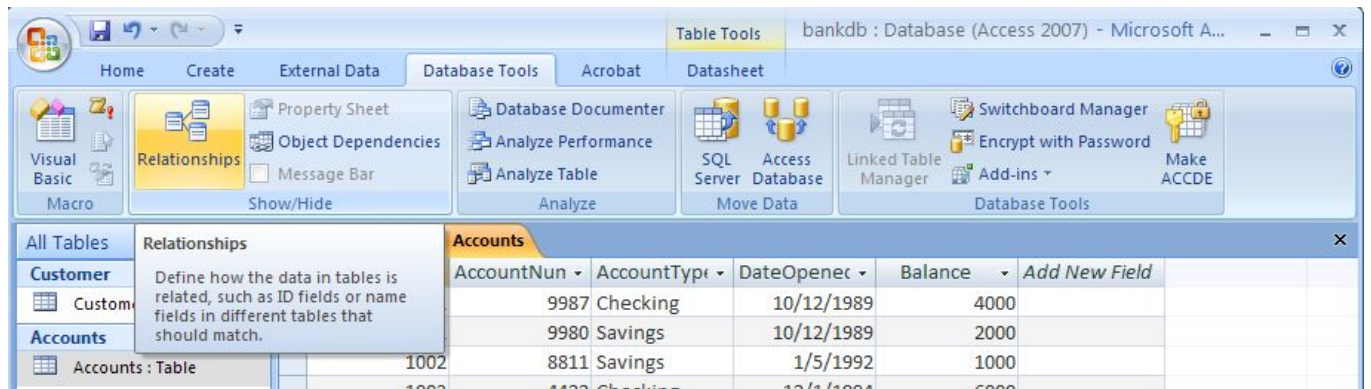
Creating Relationships Between tables

Recall that one of the main characteristics of relational databases is the fact that all tables are related to one another. In the Bank database thus far, the Customers table is related to the Accounts table by virtue of the CustomerID field appearing in both tables. Access has a means to make this relationship explicit using the Relationships screen. Access uses this information when designing reports, forms and queries that require more than one table to be displayed.

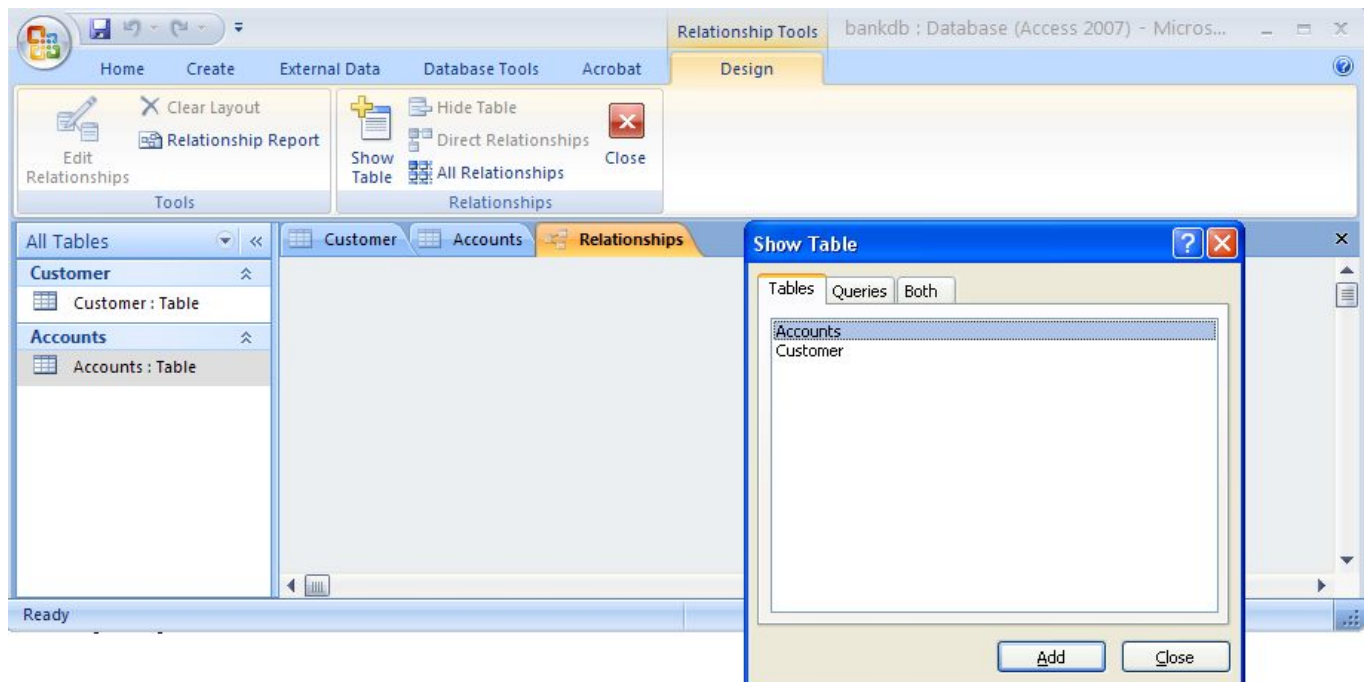
To get started, make sure the Accounts table and the Customer table are both closed. Access will halt creation of any relationships if the table are currently opened. To close a table, either right-click on the table name in the tab above the table and choose the close menu item, or click the small X to right above the table.



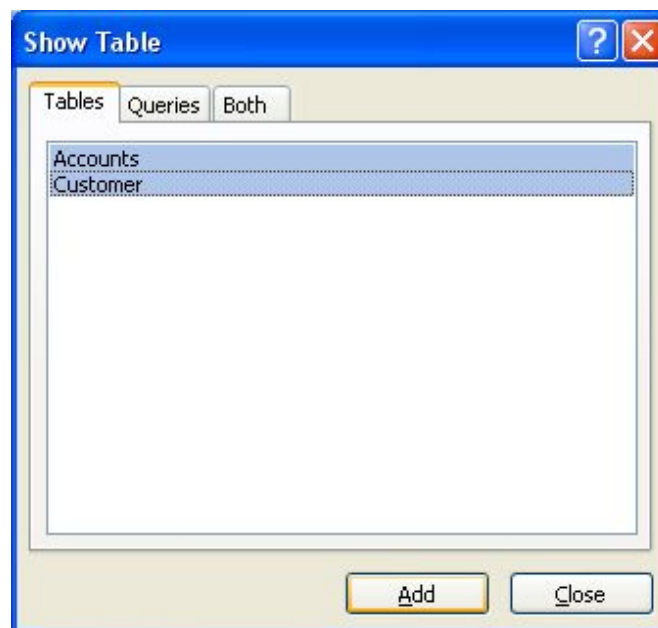
Next, display the Relationships screen by clicking on the Database Tools tab and then click on the Relationships button as shown below.



The blank Relationships screen will appear as follows:

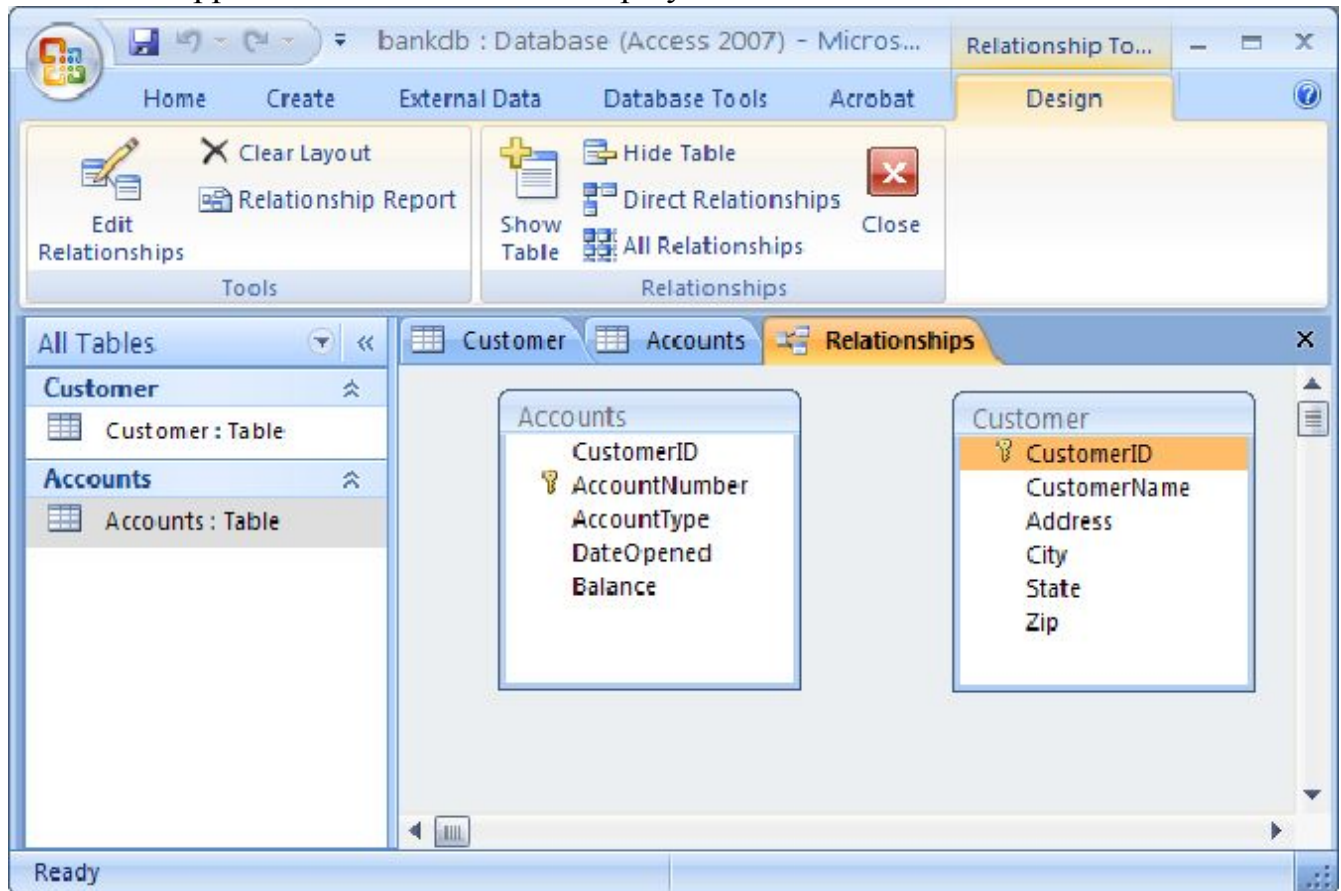


The Show Table dialog box will appear by default. Highlight both the Customers table and the Accounts table as shown below and then click on the Add button.



Then click on the Close button to close this dialog box. The Relationships screen

will now reappear with the two tables displayed as below:



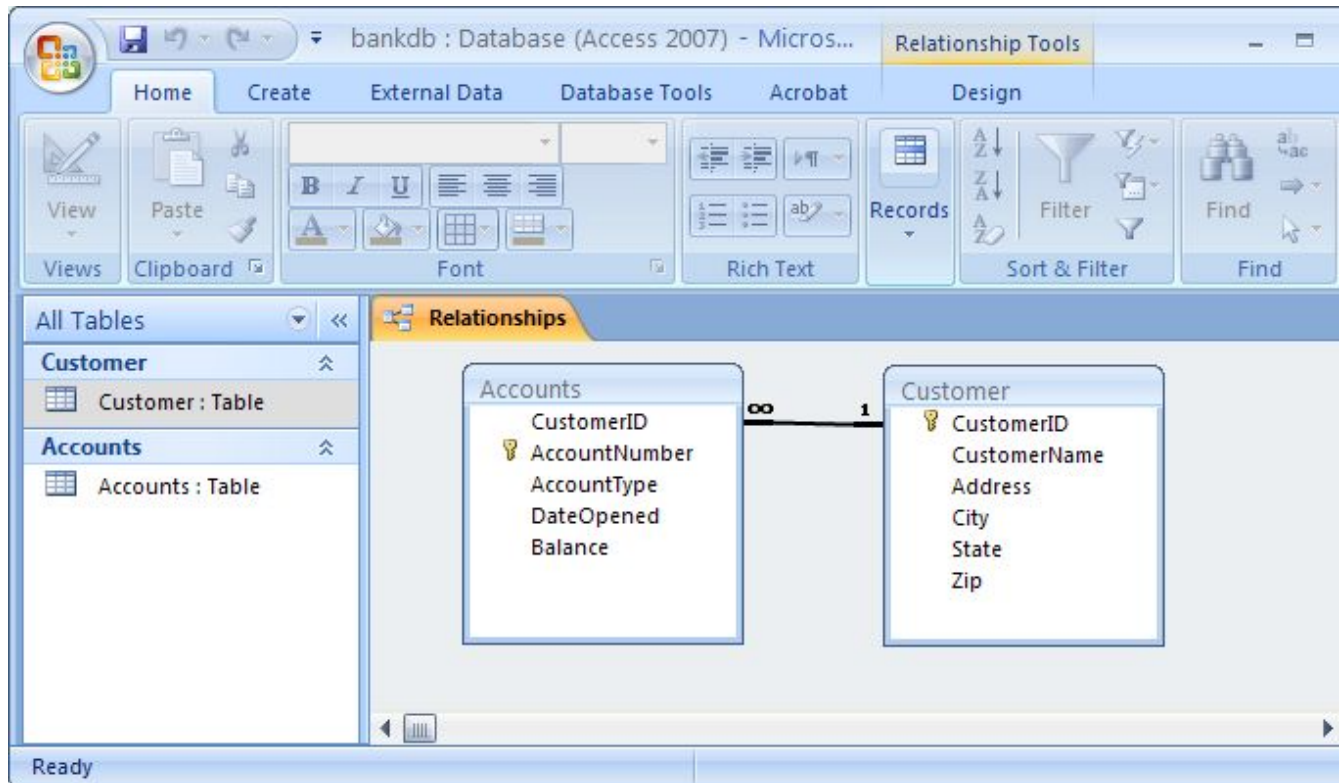
To connect the Customers table with the Accounts table to form a relationship, click on the CustomerID field in the Customers table and drag it over on top of the CustomerID field on the Accounts table. Upon releasing the mouse button, the Edit Relationships dialog box will appear as below:



Access will do its best to determine the Relationship Type (almost always it will select *One-to-Many*). For this example, Access knows that CustomerID is a key of the Customer table so it chooses this field as the "One" side. This makes the Accounts table the "Many" side as *One* customer may have *Many* accounts.

One additional step to be taken is the check off the box labeled "Enforce Referential Integrity". This option puts constraints into effect such that an Accounts record can not be created without a valid Customer record, and Access will also prevent a user from deleting a Customer record if a related Accounts record exists. At this point,

click on the `Create` button to create the relationship. The Relationships screen should reappear with the new relationship in place as follows:



Note the symbols "1" (indicating the "One" side) and the infinity symbol (indicating the "Many" side) on the relationship. Close the relationships screen and select `Yes` to save the changes to the Relationships layout.

If the relationship does not appear in the above fashion, highlight it and press the delete key to delete it. Then go back to the table design view and make certain that the `CustomerID` field is designated as the key of the `Customers` table. Then go back to the Relationships screen and try to recreate the relationship.

Review of Creating and Viewing Tables

Creating a new table requires the following steps:

1. Click on the **Tables** tab on the Access main screen
2. Click on the `New` button.
3. Choose the **Design View** and click the `OK` button.
4. Fill in the name, data type and description of each of the fields in the table.
5. Designate a primary key by clicking on one of the fields with the right mouse button and then choose `Primary Key` from the pop-up menu.
6. Save the table by pulling down the `File` menu and choosing `Save`.
7. Close the new table by pulling down the `File` menu and choosing `Close`.

To change the design of an existing table (e.g., to add, change or delete a field):

1. Click on the **Tables** tab on the Access main screen
2. Highlight the name of the table to be modified and click on the `Design` button.
3. Make the necessary changes.
4. Save the table by pulling down the `File` menu and choosing `Save`.
5. Close the table by pulling down the `File` menu and choosing `Close`.

To add, delete or change data in an existing table:

1. Click on the **Tables** tab on the Access main screen
2. Highlight the name of the table to be modified and click on the `Open` button.
3. Make the necessary changes to the data.
4. Save the table data by pulling down the `File` menu and choosing `Save`.
5. Close the table by pulling down the `File` menu and choosing `Close`.

To create or edit relationships between tables:

1. Pull down the `Tools` menu and select the `Relationships` menu item.
2. To display tables, right click and choose `Add Tables`
3. To create new relationships, drag a key field from one table and drop it on the associated field in another table
4. To edit an existing relationship, double click on the relationship line.
5. To delete an existing relationship, click on the relationship line and press the delete key.

LAB 2 CREATING QUERIES

Lab objective: to study process of creating queries

Creating and Running Queries

Queries are a fundamental means of accessing and displaying data from tables. Queries can access a single table or multiple tables. Examples of queries for our bank database might include:

- Which Customers live in Georgia?
- Which Accounts have less than a \$500 balance ?

In this section, we show how to use the Access Wizards to create queries for a single table and for multiple tables.

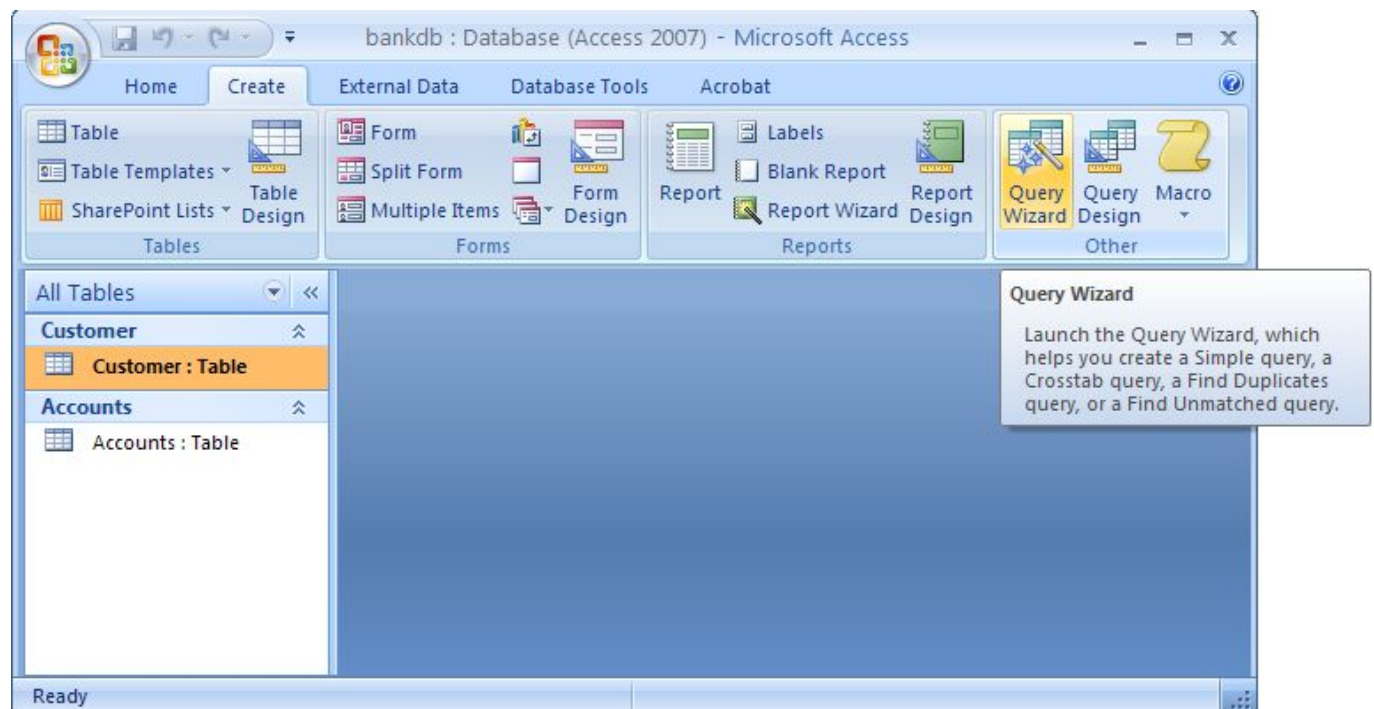
Single Table Queries

In this section, we demonstrate how to query a single table. Single table queries are useful to gain a view of the data in a table that:

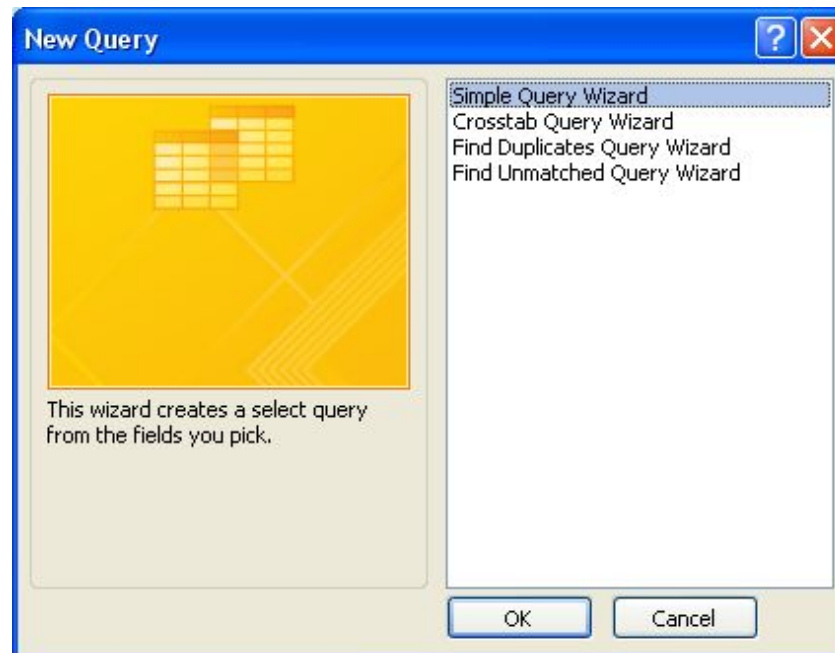
- only displays certain fields (columns) in the output
- sorts the records in a particular order
- performs some statistics on the records such as calculating the sum of data values in a column or counting the number of records, or
- filters the records by showing only those records that match some criteria. For example, show only those bank customers living in GA.

Creating a query can be accomplished by using either the query design view or the Query wizard. In the following example, we will use the query wizard to create a query.

To create a new query, click on the **Create** tab. Then click on the the **Query wizard** button.



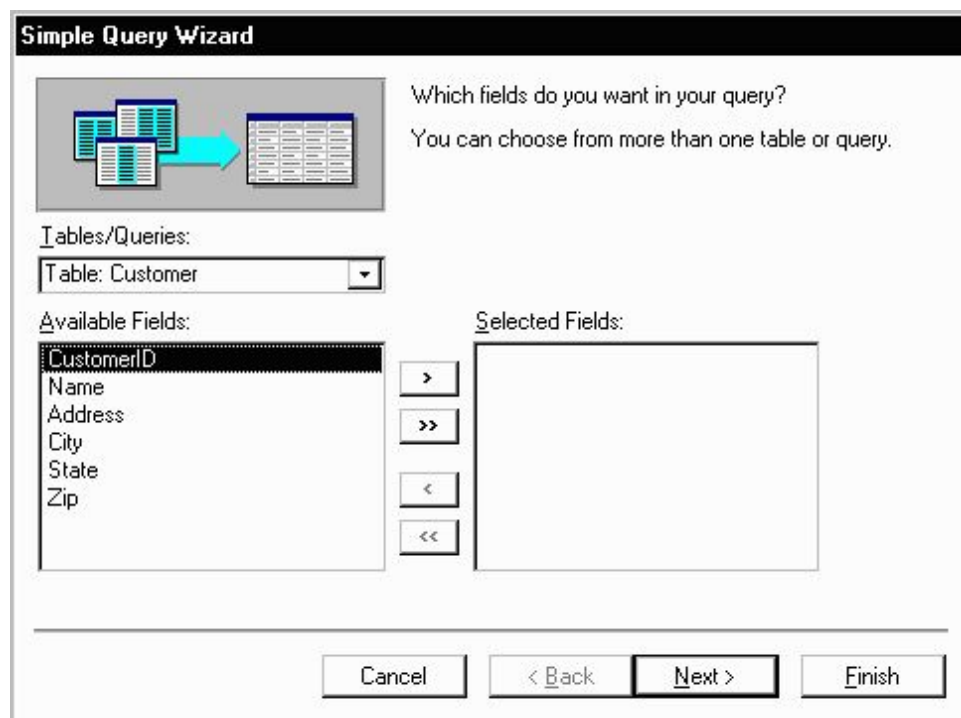
When the Query wizard appears, highlight the Simple Query Wizard selection and OK button.




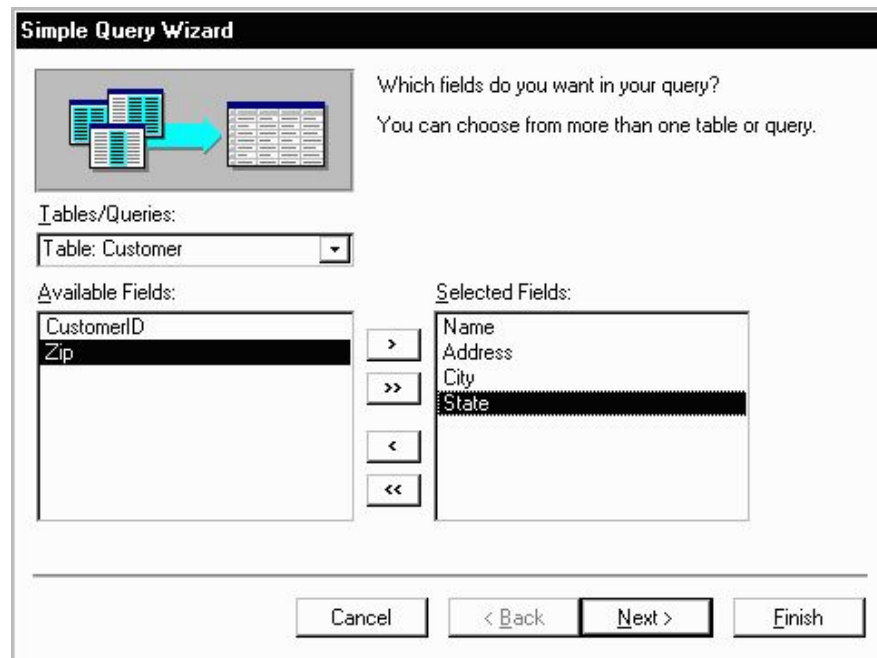
The first step in the Simple Query wizard is to specify the table for the query and which fields (columns) should be displayed in the query output. Three main sections of this step are:

1. Tables/Queries - A pick list of tables or queries you have created.
2. Available Fields - Those fields from the table that can be displayed.
3. Selected Fields - Those fields from the table that *will* be displayed.

For this example, pull down the Tables/Queries list and choose the Customer table. Notice that the available fields change to list only those fields in the Customer table. This step is shown below:



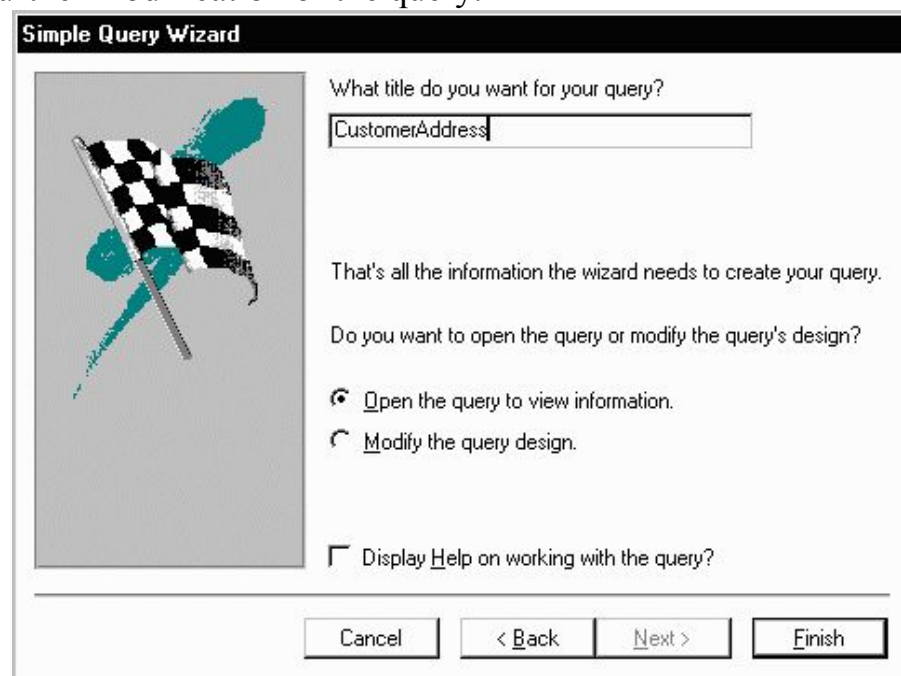
From the list of Available fields on the left, move the Name, Address, City and State fields over to the Selected Fields area on the right. Highlight one of the fields and then click on the right arrow button  in the center between the two areas. Repeat this for each of the four fields to be displayed. When done with this step, the wizard should appear as below:



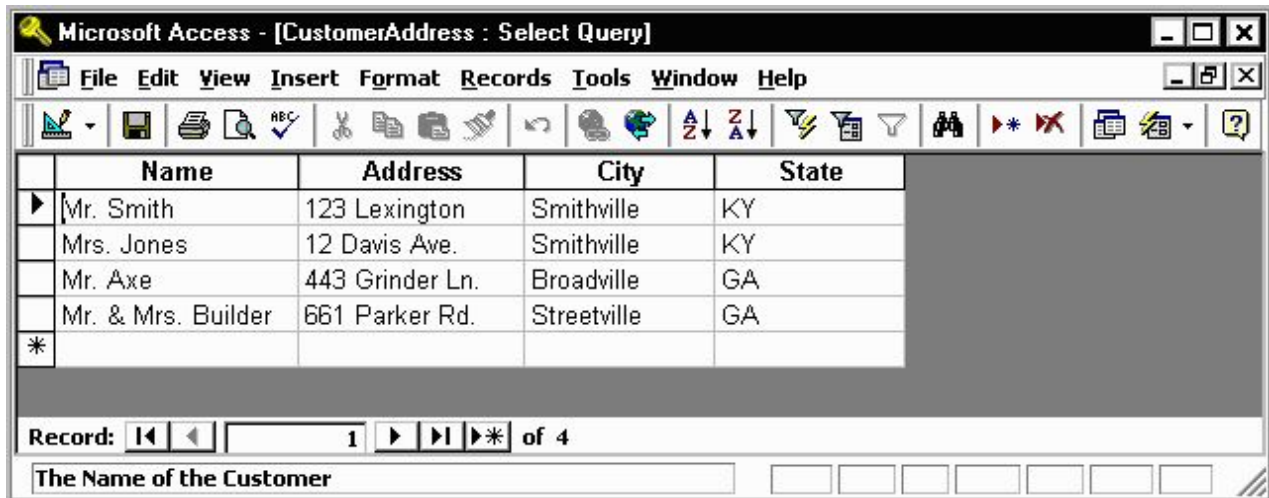
Click on the **Next** button to move to the next and final step in the Simple Query wizard. In the final step, give your new query a name. For this example, name the query: **Customer Address**

At this point, the wizard will create the new query with the option to either:

- Open the query to view information - that is, the wizard will execute the query and show the data.
- Modify the query design - the wizard will switch to the Design View to allow further modification of the query.



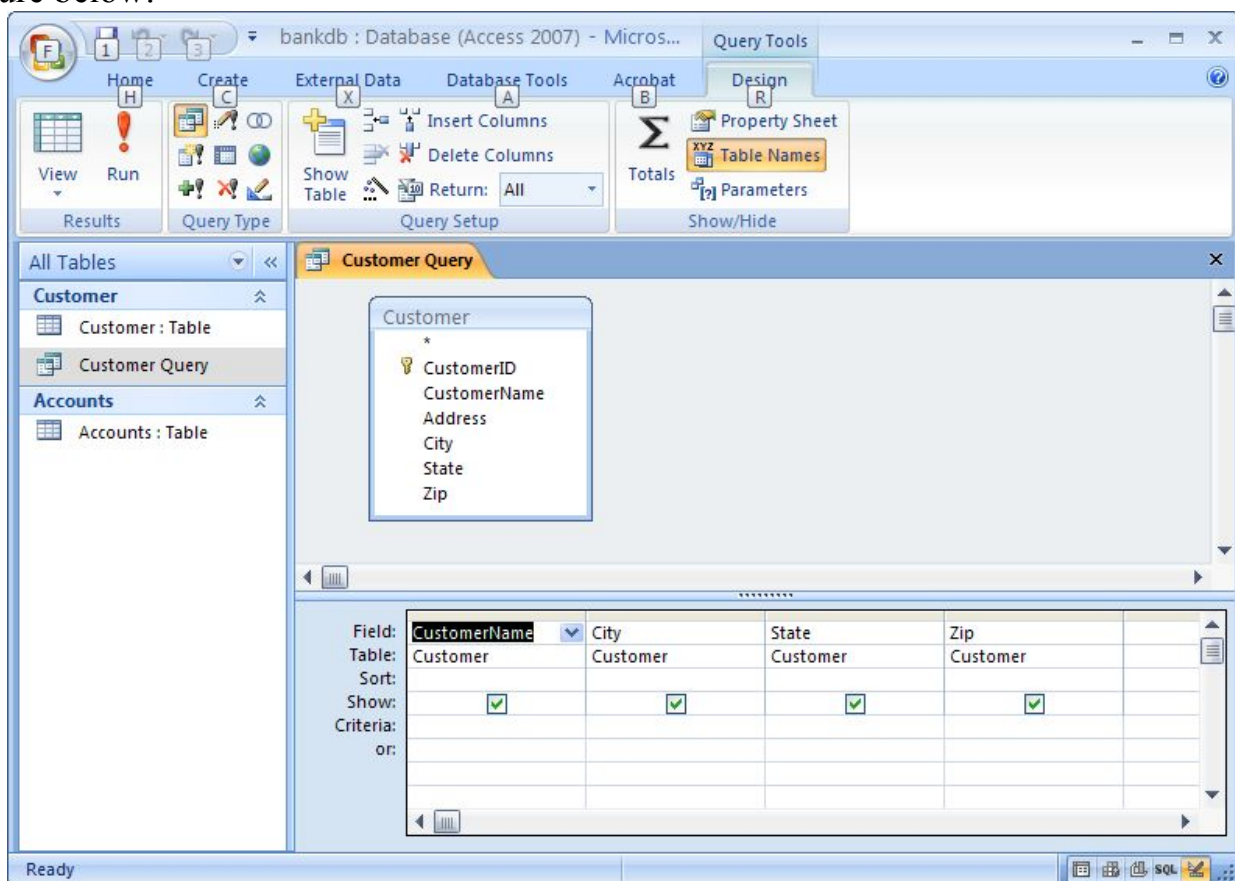
Choose **Open the query to view information** and click on the **Finish** button. When this query executes, only the customer's name, address, city and state fields appear, however, all of the rows appear as shown in the figure below:



Close this query by pulling down the `Office` menu and choosing the `Close` menu item. The Access main screen showing the `Queries` tab should appear. Note the new query `CustomerAddress` appears under the `Queries` tab.

In the following example, we will modify the `CustomerAddress` query to only display customers in a certain state. To accomplish this, we will make use of the `Query Design View`.

Open up the `CustomerAddress` query in the design view by highlighting the name of the query and clicking on the `Design` button. The design view will appear as in the figure below:



The `Query Design` view has two major sections. In the top section, the table(s) used for the query are displayed along with the available fields. In the bottom section, those fields that have been selected for use in the query are displayed.

Each field has several options associated with it:

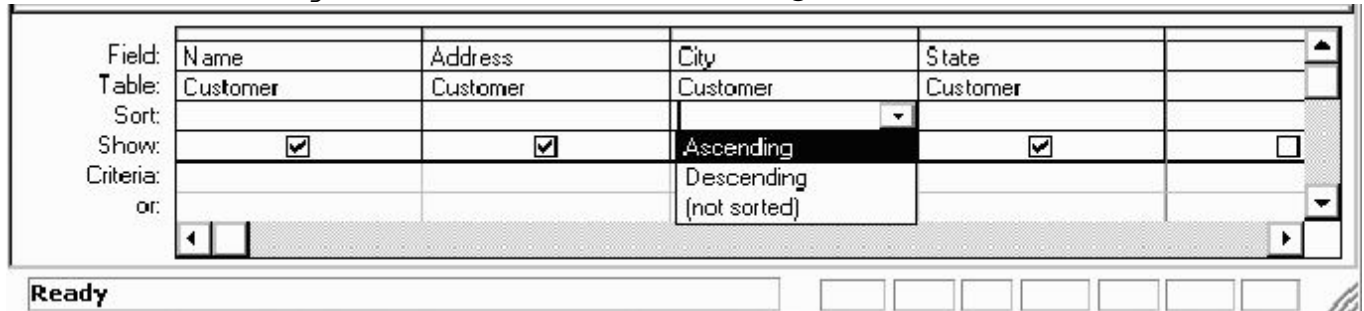
- `Field` - The name of the field from the table
- `Table` - The table the field comes from
- `Sort` - The order in which to sort on this field (`Ascending`, `Descending` or `Not`

Sorted)

- Show - Whether or not to display this field in the query output
- Criteria - Indicates how to filter the records in the query output.

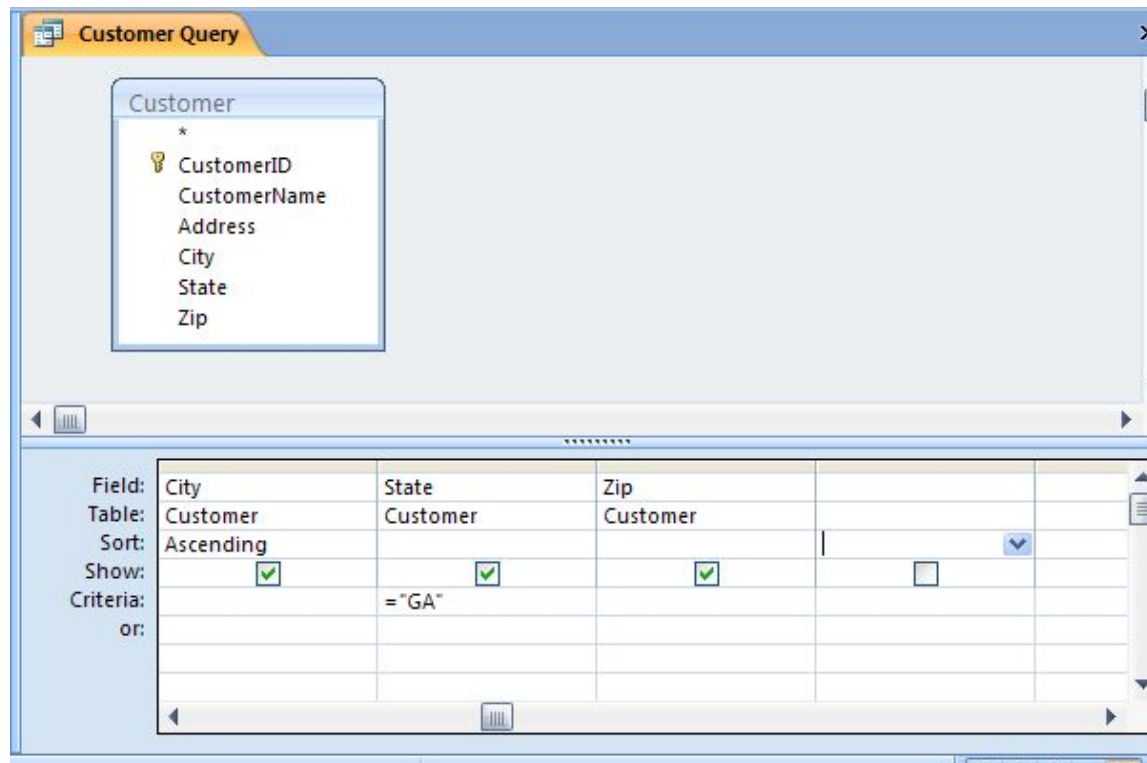
For this example, we will filter the records to only display those customers living in the State of Georgia (GA). We will also sort the records on the City field.

To sort the records on the **City** field, click in the **Sort** area beneath the **City** field. Choose **Ascending** from the list as shown in the figure below:



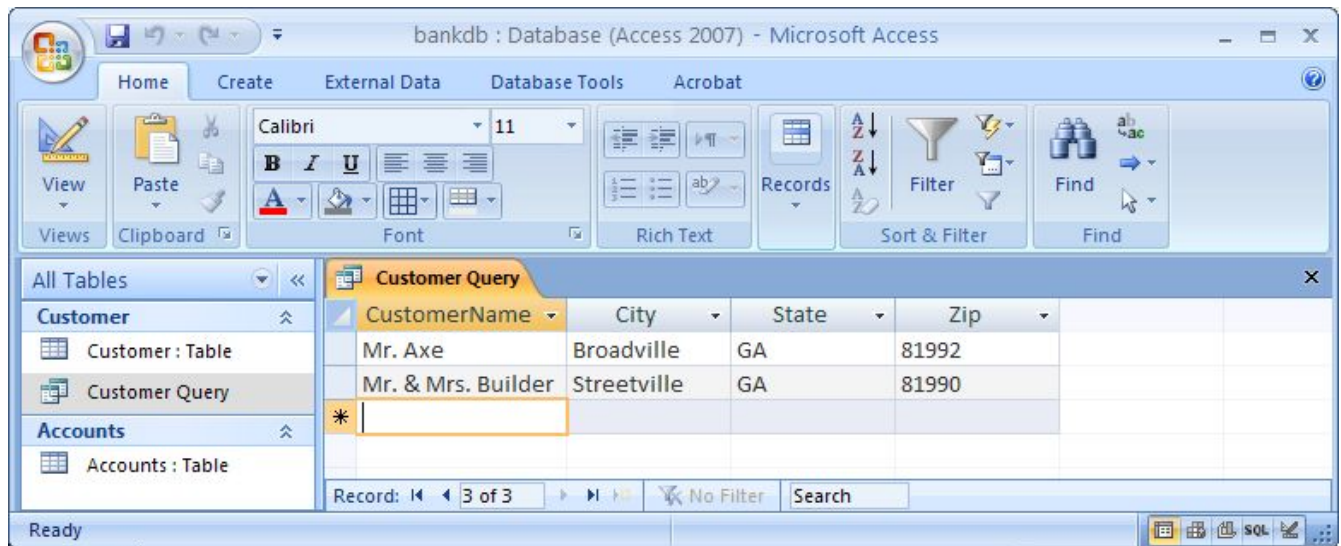
To filter the output to only display Customers in Georgia, click in the **Criteria** area beneath the **State** field and type the following statement:

= 'GA'



The = 'GA' statement tells Access to only show those records where the value of the **State** field is equal to 'GA'. Note the use of single quotes to surround the characters.

Run the query by clicking on the **Run** button (with the large red exclamation point). The output is shown in the figure below:



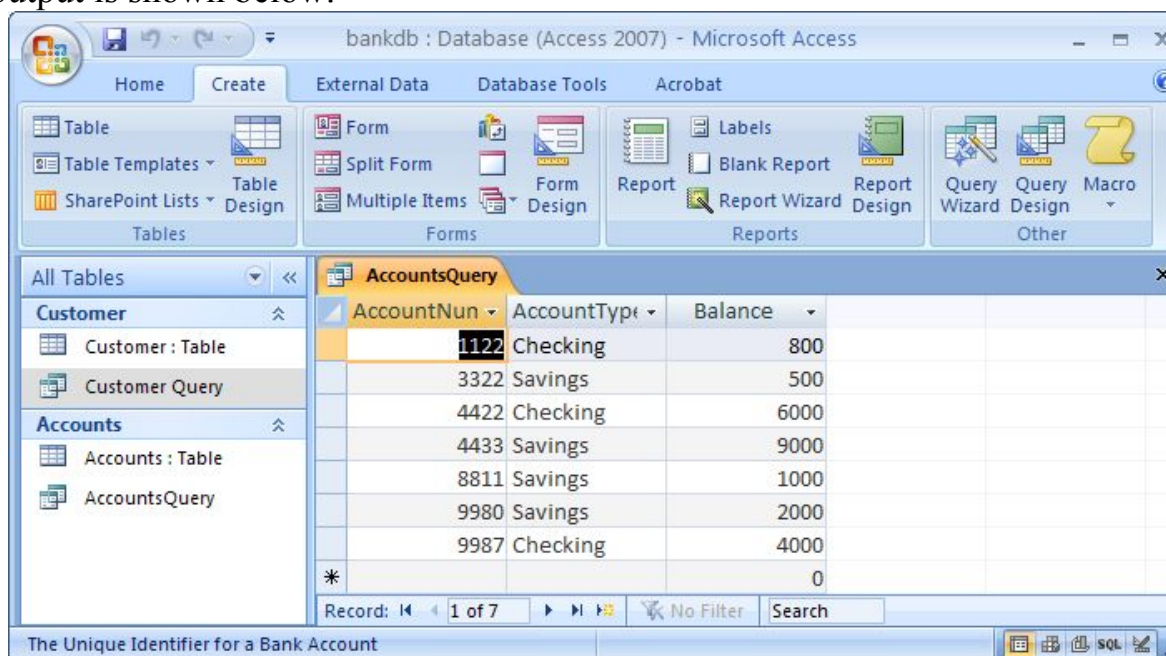
Finally, save and close this query to return to the Access main screen.

Single Table Queries

Use the Simple Query wizard to create a query on the Accounts table showing just the AccountNumber, AccountType and Balance fields.

1. From the Access main screen, click on the Queries tab. Then click on the New button.
2. Choose the Simple Query wizard option and click on the OK button.
3. Under Table/Queries: choose the Accounts table. Then move the AccountNumber, AccountType and Balance fields over to the Selected fields area. Then click the Next button.
4. In the next panel, you will be asked to choose between a detail or summary query. Choose detailed query and click on the Next button.
5. Name the new Query : AccountsQuery and click on the Finish button.

The output is shown below:



Close this query by pulling down the Office menu and choosing Close.

In the next part of the exercise, we will modify the query to sort the output on the account number and only display the Savings accounts.

1. From the Queries tab on the Access main screen, highlight the AccountsQuery and click on the Design button.
2. Change the Sort order for the AccountNumber field to Ascending.

Add the following statement to the Criteria: are under the **AccountType** field:
= 'Savings'

Field:	AccountNumber	AccountType	Balance		
Table:	Accounts	Accounts	Accounts		
Sort:	Ascending				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteria:		= 'Savings'			

Ready

- Run the query by pulling down the Query menu and choosing the Run menu item. The output is shown below:

Microsoft Access - [AccountsQuery : Select Query]

File Edit View Insert Format Records Tools Window Help

	AccountNumber	AccountType	Balance
▶	3322	Savings	500
	4433	Savings	9000
	8811	Savings	1000
	9980	Savings	2000
*	0		0

Record: 1 of 4

The Unique Identifier for a Bank Account

- Finally, save and close the query to return to the Access main screen.

LAB 3 CREATING MULTIPLE TABLE QUERIES

Lab objective: to study process of creating multiple table queries

Up to this point, queries involving only one table have been demonstrated. It is almost a given that queries will need to involve more than one table. For this example, assume that a manager would like to see a list of all of the customers and the type of account(s) that each one maintains at the bank. Such a query requires data from both the Customers table as well as the Accounts table. In such queries, Access will rely on the Relationships established between tables to guide how the data will be assembled to satisfy the query.

To start the process of creating a multiple table query, highlight the `Query` tab (Access '97) and click on the `New` button to create a new query. Select the "Simple Query Wizard" option as was done previously. When the simple query wizard appears, select the `CustomerID` and `Name` fields from the `Customers` table, then switch the `Tables/Queries` selection to the `Accounts` table and select the `CustomerID`, `AccountType` and `Balance` fields from the `Accounts` table. The result from this step is down below:

Simple Query Wizard

Which fields do you want in your query?
You can choose from more than one table or query.

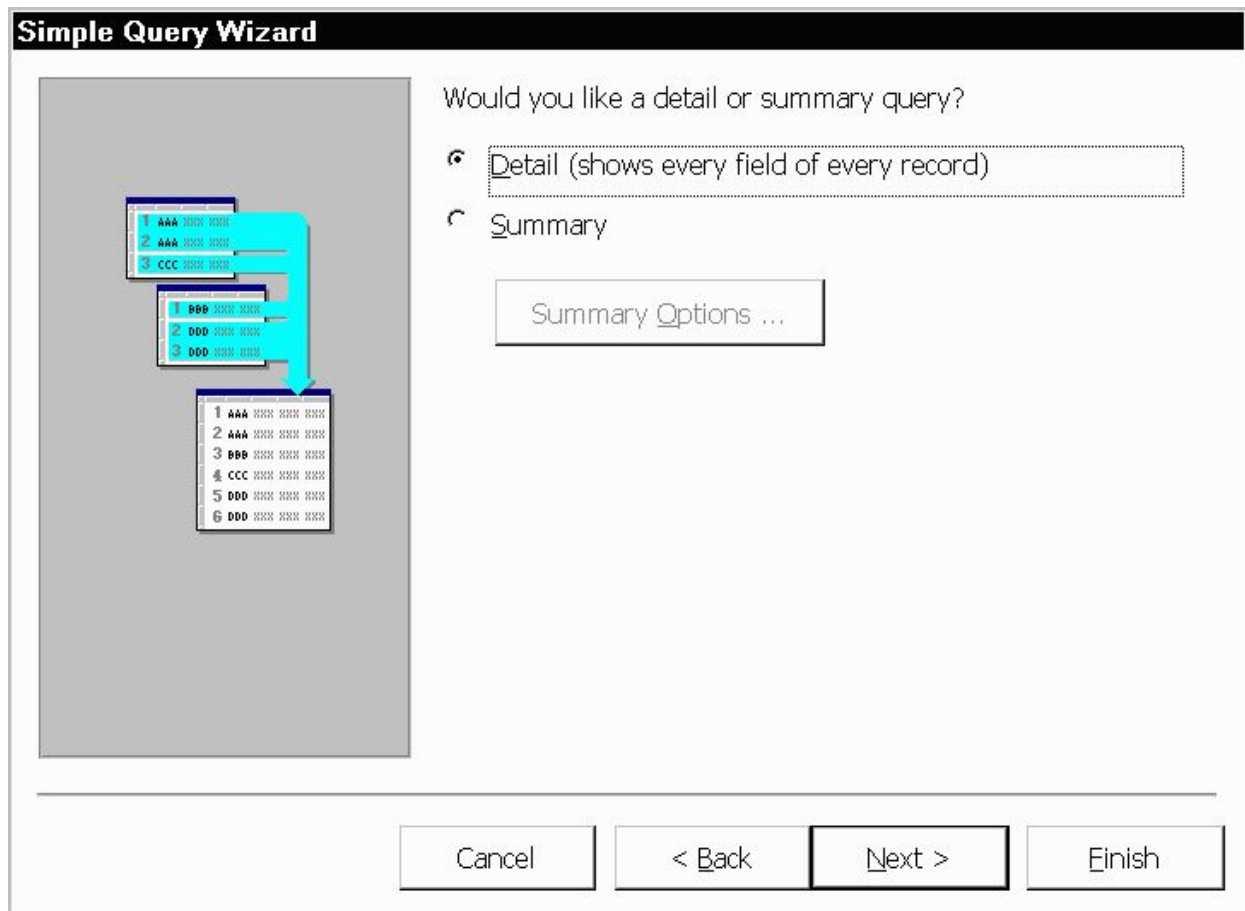
Tables/Queries:
Table: Accounts

Available Fields:
AccountNumber
DateOpened

Selected Fields:
Customer.CustomerID
Name
Accounts.CustomerID
AccountType
Balance

Cancel < Back Next > Finish

Click the `Next` button to continue. In the next step of the wizard, an option will appear to provide some level of Summary. For this example, leave the default at "Detail ..." as shown below and then click on the `Next` button.



In the final step of the wizard, name the query "Customer Accounts Query" and click on the `Finish` button. The multiple table query results should appear as follows:

	Customer_Custome	Name	Accounts_Cus	AccountType	Balance
▶	1001	Mr. Smith	1001	Savings	2000
	1001	Mr. Smith	1001	Checking	4000
	1002	Mrs. Jones	1002	Savings	1000
	1003	Mr. Axe	1003	Checking	6000
	1003	Mr. Axe	1003	Savings	9000
	1004	Mr. & Mrs. Builder	1004	Checking	800
	1004	Mr. & Mrs. Builder	1004	Savings	500
*					

Record: 1 of 7

The Unique Identifier for a Customer NUM

As with single table queries demonstrated previously, one can change the query definition in design view by adding filters (e.g., show account information for all customers in 'GA').

Multiple Table Queries

For this exercise, create a new query called "Accounts Summary Query" that joins the Customers table (include the CustomerID and Name fields) with the Accounts table (include the Balance field only). In the second step of the wizard, click on the `Summary`

choice (instead of Details) and then click on the Summary Options... button. Check off all of the Summary option boxes such as **Sum**, **AVG**, **Min** and **Max** as shown in the figure below:

What summary values would you like calculated?

Field	Sum	Avg	Min	Max
Balance	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Count records in Accounts

OK
Cancel

The resulting query should appear as follows:

CustomerID	Name	Sum Of	Avg Of Bal	Min Of B	Max Of Bal	Count Of Accounts
1001	Mr. Smith	6000	3000	2000	4000	2
1002	Mrs. Jones	1000	1000	1000	1000	1
1003	Mr. Axe	15000	7500	6000	9000	2
1004	Mr. & Mrs. Builder	1300	650	500	800	2

Record: 1 of 4
The Unique Identifier for a Customer

Review of Creating and Running Queries

In this section, the basic steps for creating and running queries were introduced. The query wizard can be used to create simple queries that access a single table. It is also possible to then modify the query to sort or filter the records.

Creating a query using the query wizard:

1. From the Access main screen, click on the Queries tab. Then click on the New button.
2. From the Queries tab on the main Access screen, click on the New button and

choose the Simple Query wizard option.

3. Under Table/Queries: choose the appropriate table for the query and then indicate which fields in the table will appear in the query output.
If querying more than one table, change the Table/Queries: selection to display additional tables and select the necessary fields.
4. If the table contains numeric fields, either detailed or summary information may be specified for the query.
5. Finally, name the new query and click on the Finish button.
As a final note, Forms and Reports can be created based on existing queries.

LAB 4 CREATING TABLE FORM

Lab objective: to study process of creating table form

Creating and Running a Data Entry Form

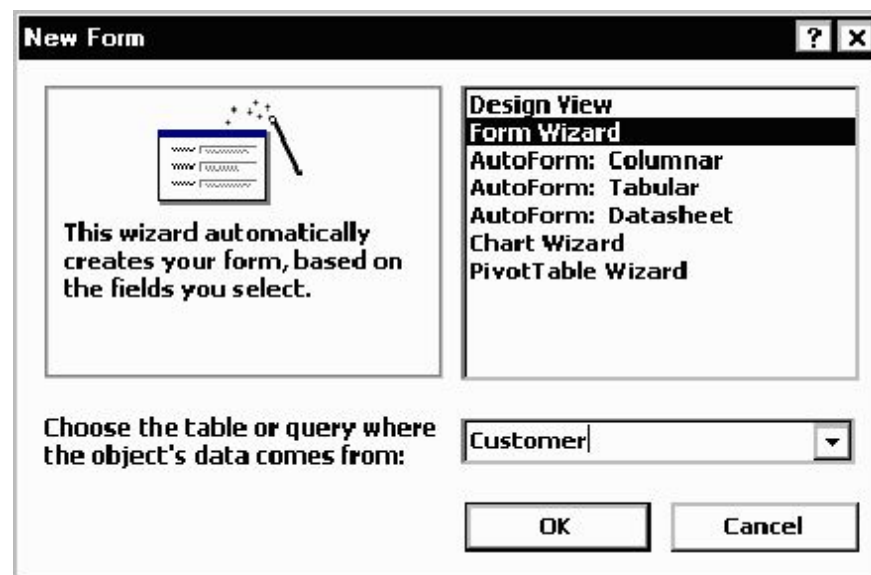
Data entry forms are the primary means of entering data into tables in the database. In a previous section, we described how to add data to a table using a spreadsheet-like view of the data. Data entry forms offer a more user-friendly interface by adding labels for each field and other helpful information.

Access provides several different ways of creating data entry forms. These include creating the forms by hand using a Design View as well as a number of wizards that walk the user through the forms creation process. In this section, we cover the basic steps for using a wizard to create a data entry form.

Creating a Single Table Form using the Wizard

In this example, we will create a simple data entry form for the Customer table. To begin the process, click on the Forms tab on the Access main screen. As with the other components in Access, there are buttons for creating a New form, Open an existing form and Design an existing form. For this example, click on the New button to create a new form.

A New Form dialog box will appear with several options for creating a new form. For this tutorial, choose the Form wizard. At the bottom of the dialog box, there is a prompt to supply the name of the table or query to be used for the new form. In this case, select the Customer table as in the following figure and then click on the OK button.



In the next step of the Form wizard, we need to specify the fields from the Customer table that will appear on the form. In this case, we want all of the fields to appear. Move each of the fields from the Available Fields side over to the Selected Fields side as in the following figure. Then click on the Next button.

Form Wizard

Which fields do you want on your form?
You can choose from more than one table or query.

Tables/Queries:
Table: Customer

Available Fields:

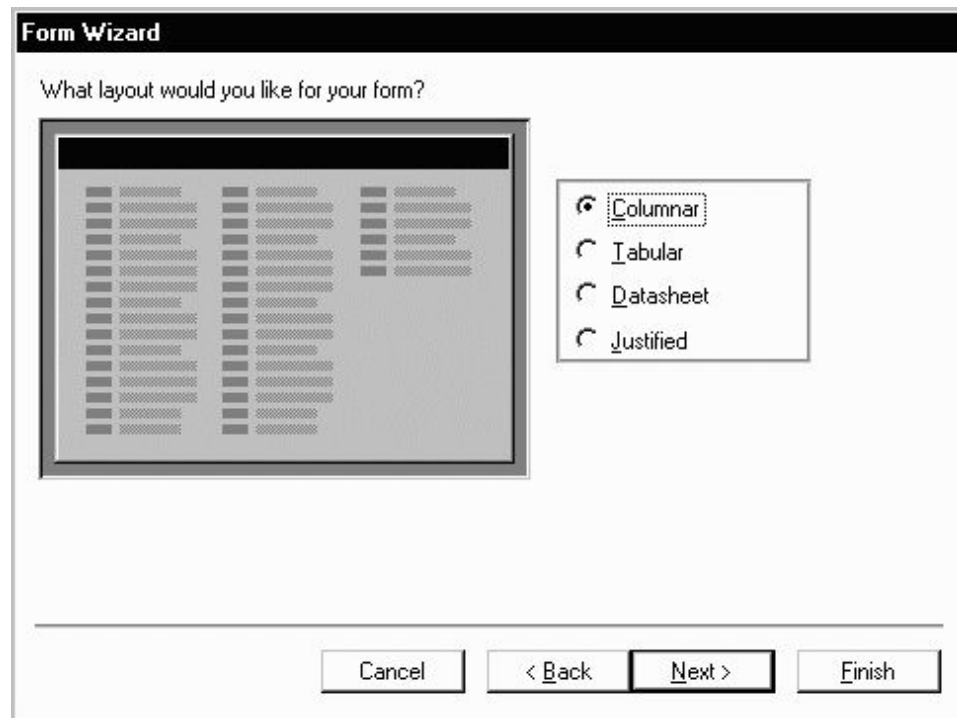
Selected Fields:
CustomerID
Name
Address
City
State
Zip

Cancel < Back Next > Finish

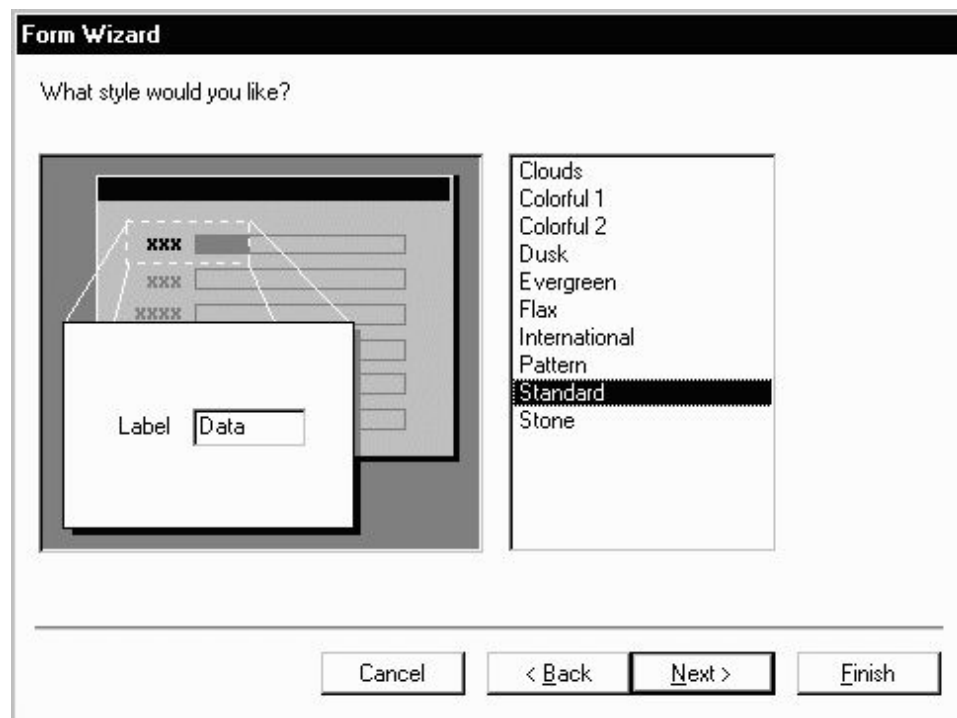
Forms can have several different layouts or arrangement of the labels and fields on the screen.

- Columnar - Places the labels to the left of each field. This is similar to a paper form. This layout is suitable for viewing data one record at a time.
- Tabular - Places the field labels at the top of the screen and the records are displayed below. This is similar to how a spreadsheet would display the data and is suitable for displaying multiple records of data at a time.
- Datasheet - The data appears in the same fashion as when viewing or adding data to a table.
- Justified - Places the labels above each field with the fields spread out on the form.

This is suitable for viewing a single record at a time as with the columnar layout. For this example, choose the columnar layout as shown in the figure below and click on the Next button.



Access has several sample display styles that determine how the form will appear, including elements such as fonts, colors and the background used in the form. For this example, select the `Standard` style as shown below and click on the `Next` button.



As a final step, give this new form the name: `CustomerDataEntry` and then click on the `Finish` button as shown below:

The new form will be created by the wizard and then opened. It should appear as in the figure below:

Use the tab key to navigate between fields in the form. To move to the next or previous record, use the record navigation bar at the bottom of the form:

Record: of 4

The buttons on the navigation bar perform the following functions:

- Go to the first record.
- Go to the previous record.
- Go to the next record.
- Go to the last record.
- Go past the last record to add a new record.

To close the form and return to the Access main screen, pull down the **File** menu and choose **Close**.

To open the form at any time, highlight the form name under the **Forms** tab on the Access main screen and click on the **Open** button.

Exercise: Creating a Single Table Form

For this exercise, we will create a data entry form for the Accounts table created in a previous exercise.

1. Click on the **Forms** tab on the Access main screen and then click on the **New** button to create a new form.
2. Select the **Form wizard** and select the **Accounts** table. Then click the **OK** button.
3. Select all of the available fields and click on the **Next** button.
4. Choose a **Tabular** layout and click on the **Next** button.
5. Choose the **Standard** style and click on the **Next** button.
6. Name the form: **AccountsDataEntry**

Then click on the **Finish** button to create, save and view the new form.

The new form is shown in the figure below:

CustomerID	AccountNumber	AccountType	DateOpened	Balance
1004	1122	Checking	11/13/88	800
1004	3322	Savings	8/22/94	500
1003	4422	Checking	12/1/94	6000
1003	4433	Savings	12/1/94	9000
1002	8811	Savings	1/5/92	1000
1001	9980	Savings	10/12/89	2000
1001	9987	Checking	10/12/89	4000
*	0	0		0

Record: 1 of 7

The Unique identifier for a customer

Close the form and return to the Access main screen, by pulling down the **File** menu and choosing **Close**.

Review of Creating and Running a Data Entry Form

The basic steps for creating a simple data entry form are:

1. Choose a table and a form wizard
2. Specify the fields (columns) that will appear in the form
3. Specify the layout for the form
4. Specify the style (fonts/colors, etc.) for the form
5. Save, create and run the new form

In this section we covered the basic steps required to create and run a data entry form. Access provides wizards which are adept at building simple forms with a minimal amount of work. More advanced work on forms would concentrate on using the Design View to change a form's appearance and to add or remove fields and labels once a form is created.

LAB 5 CREATING TABLE REPORT

Lab objective: to study process of creating table report

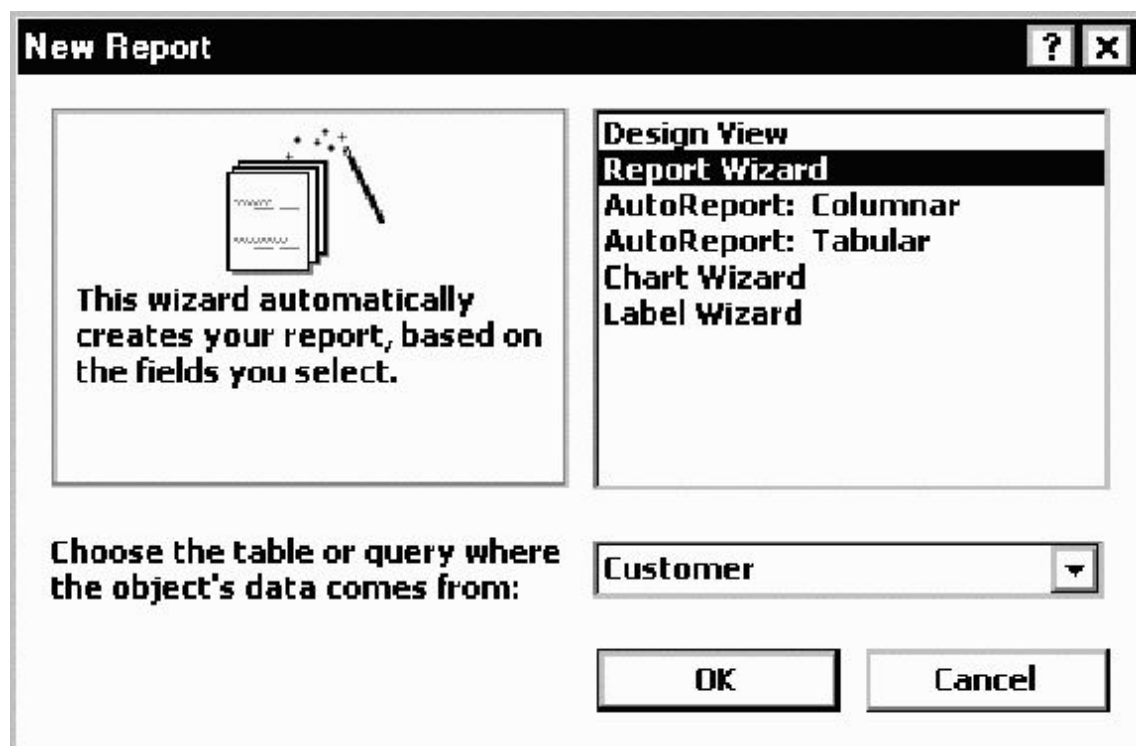
Creating and Running a Report

Reports are similar to queries in that they retrieve data from one or more tables and display the records. Unlike queries, however, reports add formatting to the output including fonts, colours, backgrounds and other features. Reports are often printed out on paper rather than just viewed on the screen. In this section, we cover how to create simple reports using the Report wizard.

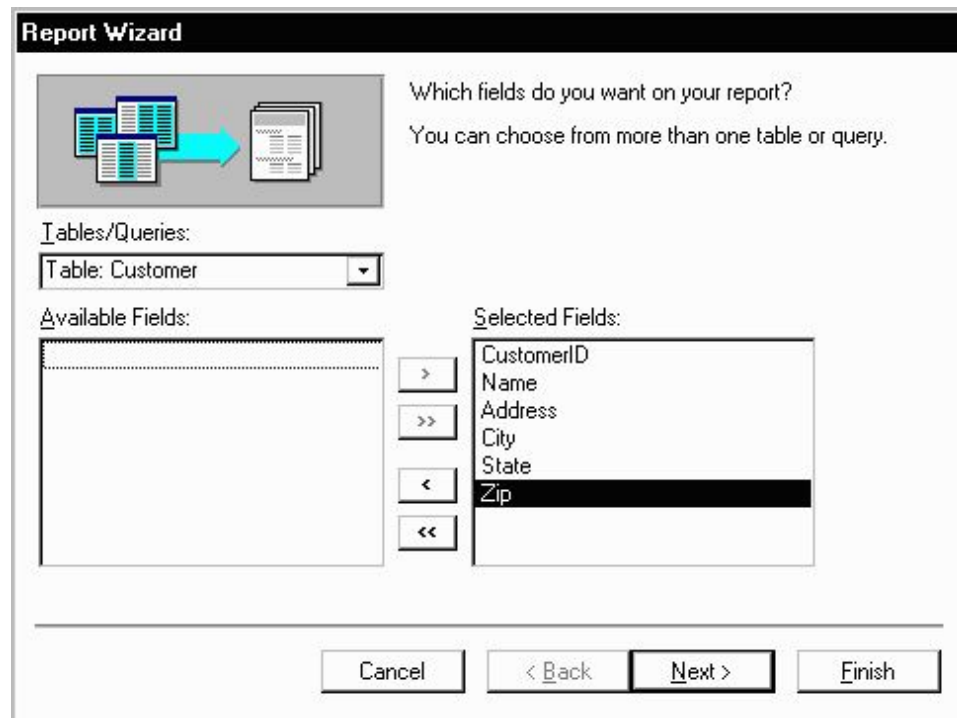
Creating a Single Table Report using the Wizard

In this example, we will create a simple report for a single table using the Report wizard. As with the Queries and Forms, we begin by selecting the Reports tab from the Access main screen.

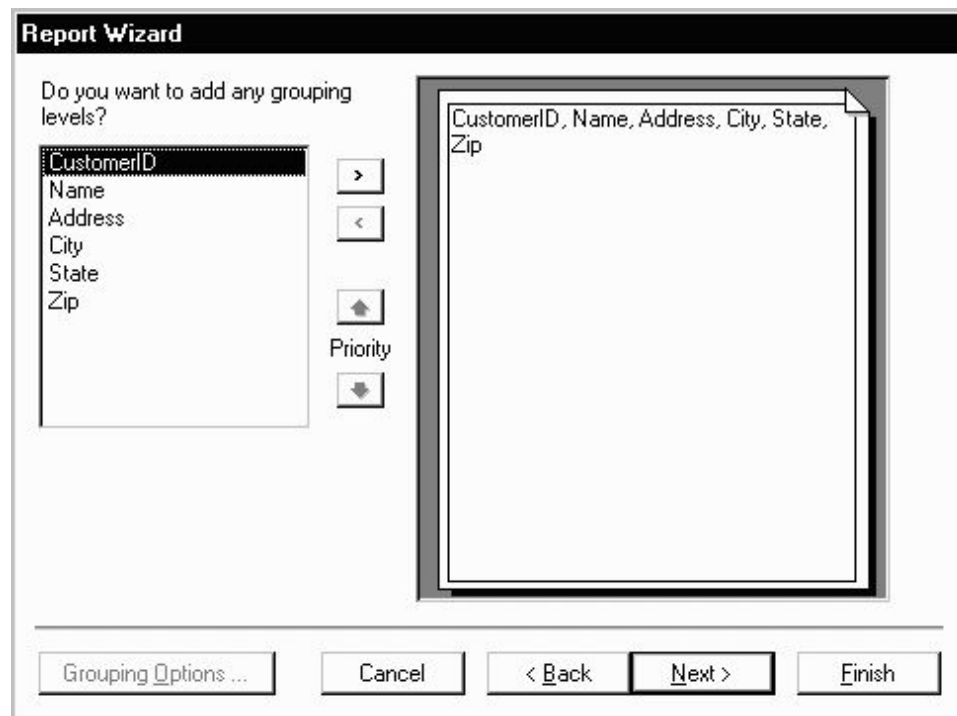
To create a new report, click on the New button. The New Report dialog box will appear as shown below. Select the Report wizard and then select the Customer table as shown below. Then click the OK button.



In the next step of the Report wizard, we need to specify the fields from the Customer table that will appear on the report. In this case, we want all of the fields to appear. Move each of the fields from the Available Fields side over to the Selected Fields side as in the following figure. Then click on the Next button.



In the next step, we have the opportunity to add *Grouping Levels* to the report. A grouping level is where several records have the same value for a given field and we only display the value for the first records. In this case, we will not use any grouping levels so simply click on the `Next` button as shown below.



In the next step, we are given the opportunity to specify the sorting order of the report. For this example, we will sort the records on the `CustomerID` field. To achieve this, pull down the list box next to the number 1 : and choose the `CustomerID` field as shown in the figure below. Then click on the `Next` button.

Report Wizard

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 CustomerID [A-Z ↓]

2 [] [A-Z ↓]

3 [] [A-Z ↓]

4 [] [A-Z ↓]

Cancel < Back Next > Finish

The next step is to specify the layout of the report. The three options are:

- Columnar - Places the labels to the left of each field. This is similar to a paper form.
- Tabular - Places the field labels at the top of the report page and the records are displayed below. This is similar to how a spreadsheet would display the data.
- Justified - Places the labels above each field with the fields spread out on the report page.

Generally, reports use the tabular layout. For this example, choose Tabular layout and set the page Orientation to Landscape so that all of the fields will fit across one page. This is shown in the figure below. Click on the Next button to continue.

Report Wizard

How would you like to lay out your report?

Layout

Columnar

Tabular

Justified

Orientation

Portrait

Landscape

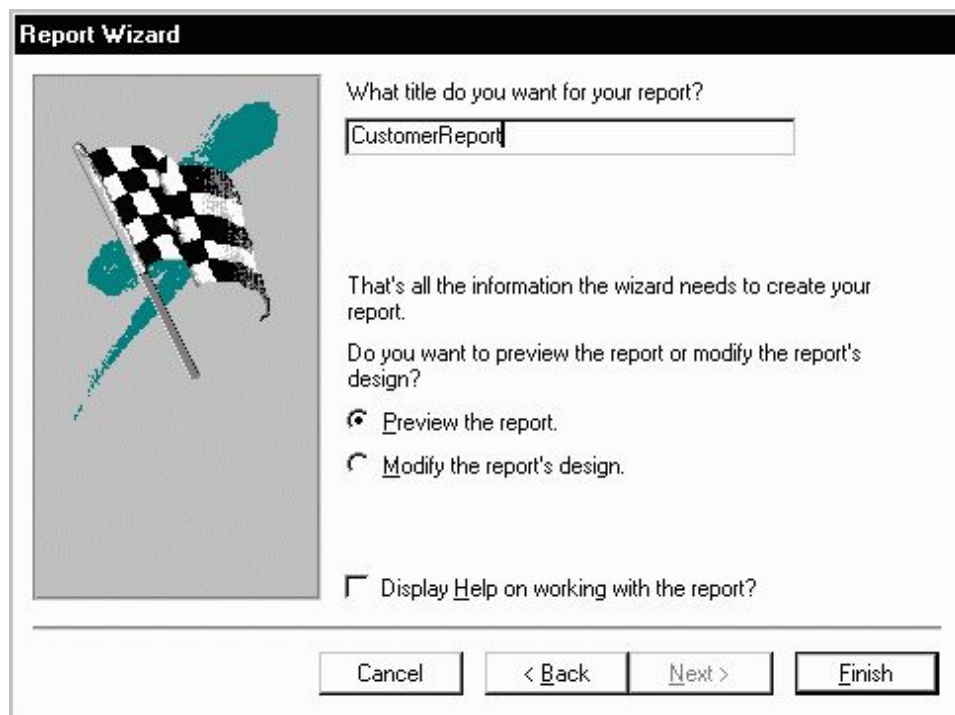
Adjust the field width so all fields fit on a page.

Cancel < Back Next > Finish

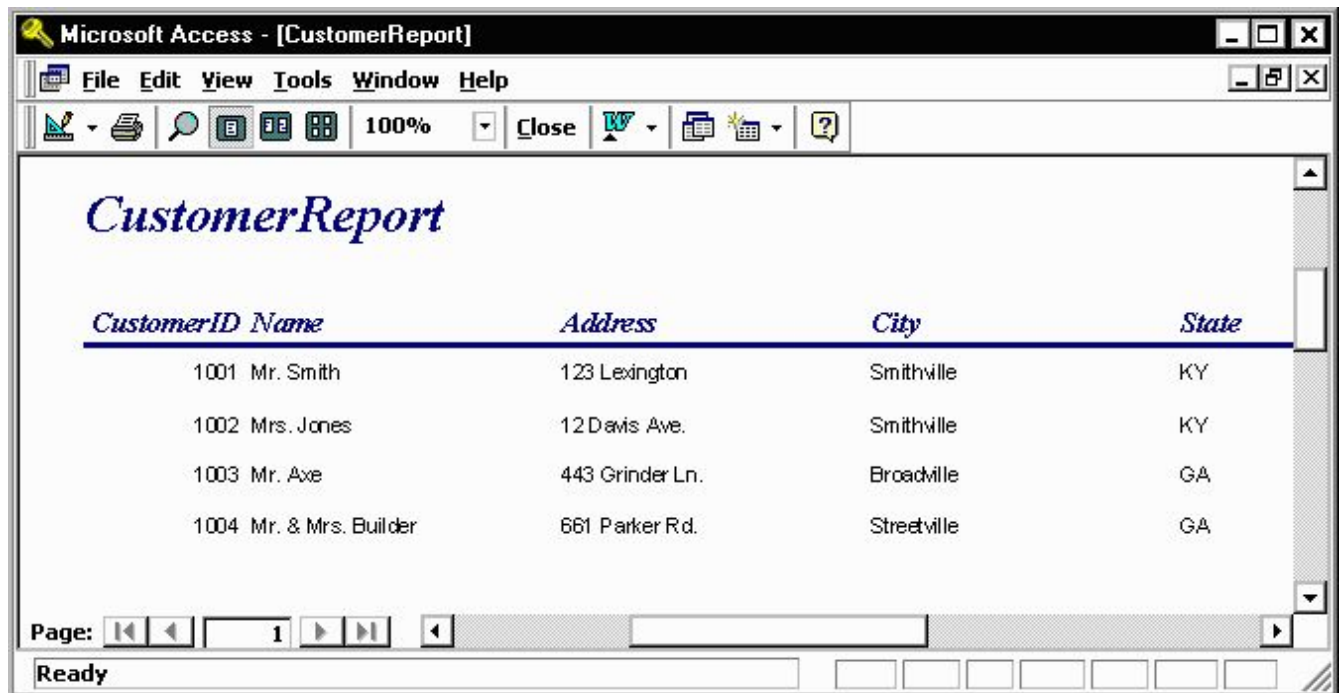
In the next step, the style of the report can be selected. For this example, choose the Corporate style and click on the Next button to continue.









Finally, give a name for the new report: `CustomerReport` and then click on the `Finish` button to create, save and display the new report.



The output from the report is shown in the figure below. Note that on some screens, the last field, `Zip`, may not display without scrolling over to the right.




Once the report is displayed, it can be viewed, printed or transferred into Microsoft Word or Microsoft Excel. The button bar across the top of the screen has the following functions:

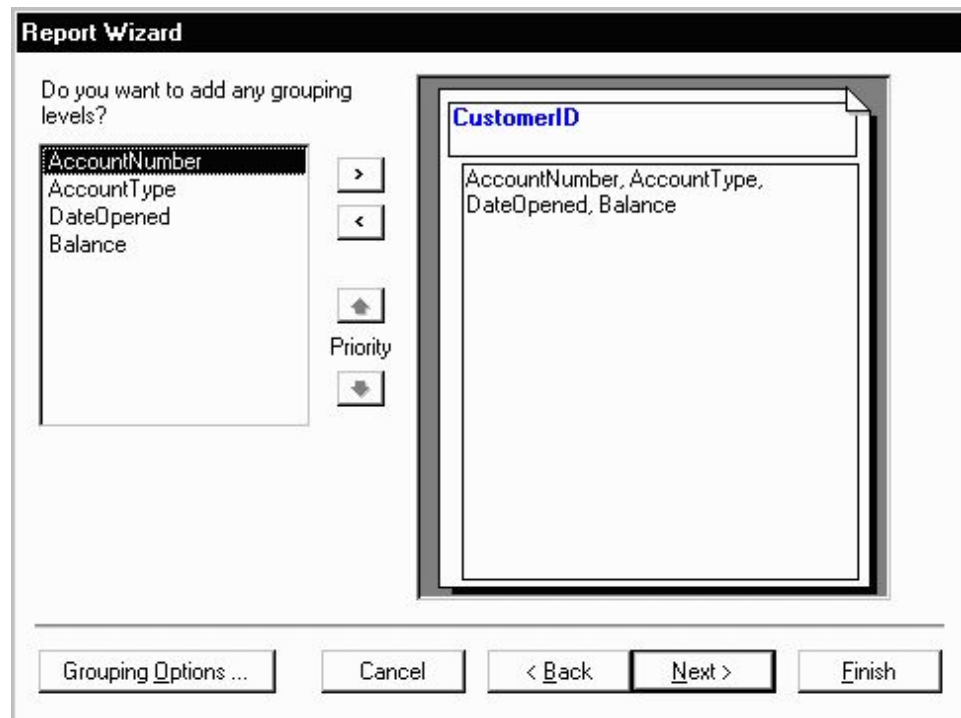
	Print the report
	Zoom into a region of the report
	Display the report as one, two or multiple pages
	Zoom into or out of the report
	Transfer the report into MS Word
	Close the report

To close the report and return to the Access main screen, pull down the **File** menu and choose **Close** or click on the **Close** button.

Exercise: Creating a Single Table Report

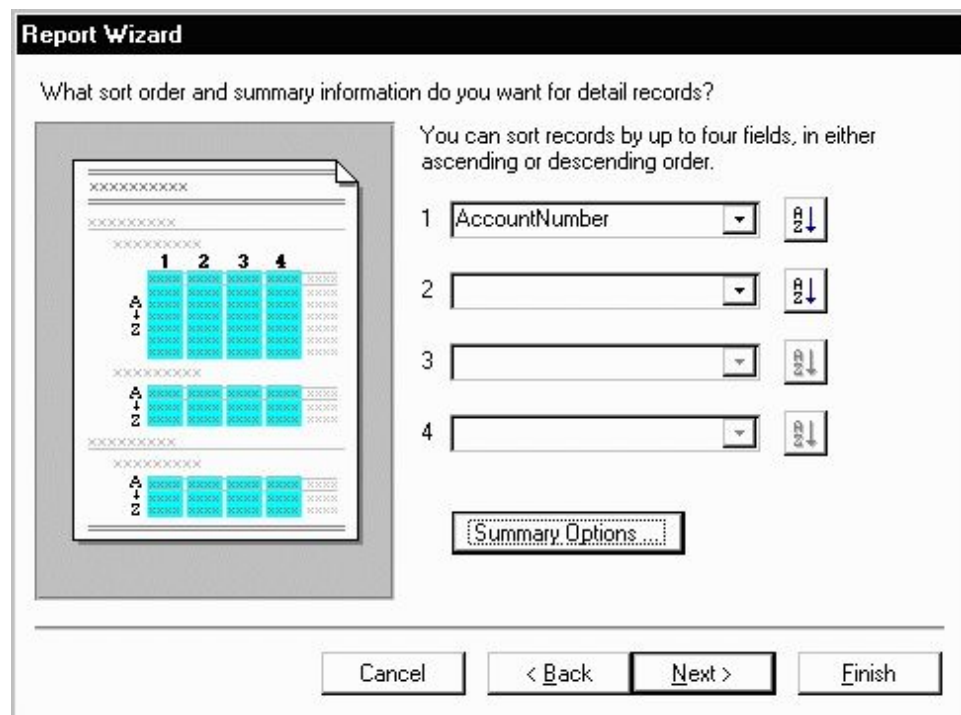
For this exercise, we will create a report showing all of the Accounts information.

1. From the Reports tab on the Access main screen, click on the **New** button.
2. Select the Report wizard, select the Accounts table and then click the **OK** button.
3. Select all of the fields in the Accounts table by moving them all over to the **Selected Fields** side and then click **Next**.
4. Group the report by CustomerID by clicking on the CustomerID field and then clicking on the right arrow  button. This is shown in the following figure:



Click on the Next button.

5. Choose to sort the report on the AccountNumber field. Note that a new button will appear called Summary Options.



Click on the Summary Options button. Choose the Balance field and select the Sum option. Choose the option to show both Detail and Summary data. Then click on the OK button.

Summary Options

What summary values would you like calculated?

Field	Sum	Avg	Min	Max
Balance	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

OK
Cancel

Show

Detail and Summary
 Summary Only

Calculate percent of total for sums

Click on the Next button.

6. Choose a Block layout and click on the Next button.
7. Choose the Corporate style and the click on the Next button.
8. Finally, name the report: AccountsReport and click on the Finish button to create, save and run the report.

The output from the AccountsReport is shown below:

Microsoft Access - [AccountsReport]

File Edit View Tools Window Help

100% Close

AccountsReport

CustomerID	AccountNumber	AccountType	DateOpened	Balance
1001	9980	Savings	10/12/89	2000
	9987	Checking	10/12/89	4000
Summary for 'CustomerID' = 1001 (2 detail records)				
Sum				6000
1002	8811	Savings	1/5/92	1000
	Summary for 'CustomerID' = 1002 (1 detail record)			
Sum				1000
1003	4422	Checking	12/1/94	6000
	4433	Savings	12/1/94	9000
Summary for 'CustomerID' = 1003 (2 detail records)				
Sum				15000
1004	1122	Checking	11/13/88	800
	3322	Savings	8/22/94	500
Summary for 'CustomerID' = 1004 (2 detail records)				
Sum				1300
Grand Total				23300

Page: 1

Ready

Note the Grouping at the level of the CustomerID and the Sum for each customer's balances.

To close the report and return to the Access main screen, pull down the File menu and choose Close.

Review of Creating and Running a Report

As can be seen in the report exercise, there are many ways to create reports to show summarization, sorting and layout of the data. Further study of Reports will show how to modify the layout using the Design View. Students are encouraged to work with the Report wizards to create different styles and types of reports.

In this tutorial, we have covered the basics for creating an Access database including tables with data, queries to retrieve data, forms to enter data and reports to display and summarize data.

Students are encouraged to further their Access knowledge and skills by working through more advanced tutorials and by reading the on-line help and Access documentation.

LAB 6 CREATING MACROS

Lab objective: to study process of creating macros

If you find yourself doing the same routine task over and over again, you might want to consider creating a macro to complete the task for you. A macro helps you perform routine tasks by automating them. Instead of manually performing a series of time-consuming, repetitive actions, you can record a single macro that does the entire task all at once for you. For example, instead of clicking the Reports icon in the Objects bar in the database window, finding and opening a specific report, printing it, and then closing it, you could create a macro to print the report with the click of a single button.

A macro is a set of one or more actions that perform a particular operation, such as opening a form or printing a report. Macros can help you to automate common tasks. For example, you can run a macro that prints a report when a user clicks a command button.

In a way, you can think of macros as a very simple introduction to programming because you can use them to create automated tasks and somewhat complex procedures. Best of all, you don't have to know a single line of code Access provides you with everything you need to write a macro.

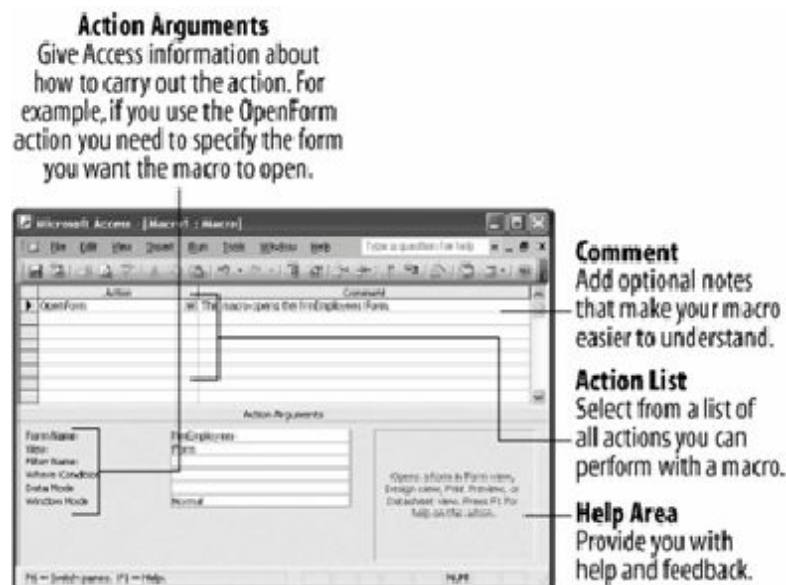


Figure 1. You create macros

In some programs, such as Microsoft Excel or Word, you can create macros with a "macro recorder" to record your commands, keystrokes, and mouse clicks. Unfortunately, there isn't a "macro recorder" or Macro Wizard to help you create a macro in Microsoft Access. Instead, you create macros by entering the actions and arguments directly in Macro Design view, shown in Figure 1. Don't worry it's not as difficult as it sounds. Working in Macro Design view really isn't all that different from working in Table Design view it's where you define and edit your macro objects.

Simple macros that automate a single task, such as opening a form or report, are incredibly simple to create we'll create such a macro in this lesson. More complicated macros with several steps or procedures may require a little bit of planning. Before you create a complicated macro, think about what you want the macro to do and the

individual actions that are required to complete this operation. Practice the steps needed to carry out the operation and write them down as you go it will make writing the macro a lot easier.

And so, without any further ado, let's create our first macro.

1. Start Microsoft Access, open the Lesson 10 database, click the Macros icon in the Objects bar, and click the New button.

The Macro1: Macro window appears, as shown in Figure 1. The Action cell is where you tell Access what you want the macro to do.

2. Click the first blank Action cell, then click the list arrow.

A list of actions appears. An action, or command, is the basic building block of a macro it's an instruction that tells Access what you want the macro to do. There are more than 50 different actions you can choose from. When you start creating your own macros you will almost certainly want to refer to Appendix to help you find the right action.

3. Scroll down and select the OpenForm action.

The OpenForm action is added to the first line of the macro window. Most of the time you will have to give Access more information about how to execute each action. For example, here we will have to tell Access which form to open with the OpenForm action. You use arguments to supply Microsoft Access with information about how to carry out the action. Each type of action has its own set of arguments, which appear in the Action Arguments panel, located at the bottom of the macro window.

4. Click the Form Name text box in the Action Arguments panel, click the list arrow, and select frmEmployees.

That's the only argument we need to specify for this exercise, but notice that there are additional arguments for the OpenForm action, such as the View argument, which lets you select the view in which to open the form (Form view, Design view, or Print Preview).

If you want, you can type a comment to explain the action in the Comment column. If you've ever had any programming experience, the Comment column is the same as a remark statement.

5. Click the first blank Comment cell and type This macro opens the frmEmployees Form.

Comments are completely optional, but they do make your macros easier to understand, especially if other users will edit them.

6. Click the Save button on the toolbar, save the macro as mcrEmployees and click OK.

You're finished working in the Macro window for now so...

7. Close the Macro window.

Time to test your new macro. The Macros icon in the Objects bar in the Database window should be selected.

8. Double-click the mcrEmployees macro.

Access runs the mcrEmployees macro and opens the frmEmployees form.

9. Close the frmEmployees form.

Editing a Macro

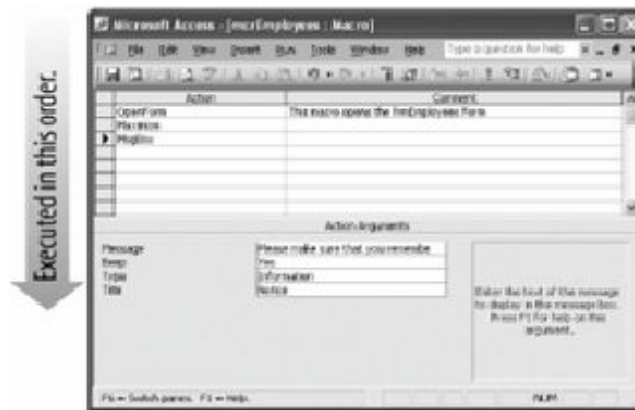


Figure 2. The mcrEmployees macro with two additional actions

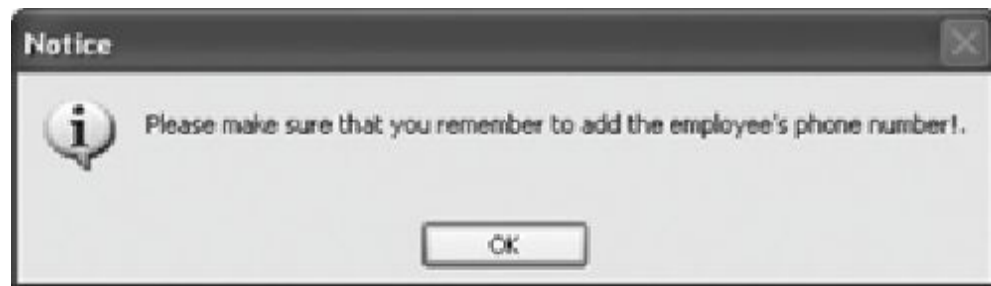


Figure 3. A message box the result of the MsgBox action

Some Microsoft Access tasks require several steps. For example, a particular task might require you to (1) open a form, (2) select a specific record, (3) select a specific field in that record, and then (4) copy the information in that field to the Windows clipboard. Macros can contain as many actions as necessary to automate even the most complicated tasks. Each action appears in its own row and is evaluated and executed in the order in which it appears in the Macro window, from top to bottom.

In this lesson you will edit the macro you created in the previous lesson to change its arguments and add some more steps or actions.

1. Select the mcrEmployees macro and click the Design button.

The mcrEmployees macro opens in Design view. Let's add two more actions to this macro.

2. Click the Action cell just below the OpenForm action, click the list arrow, scroll down the list, and select Maximize.

When you run the macro, the Maximize action will maximize the window so that it fills the entire Microsoft Access window. Because the Maximize action is so simple and straightforward, it doesn't have any additional arguments.

Let's add another action to the mcrEmployees macro.

3. Click the Action cell just below the Maximize action, click the list arrow, scroll down, and select MsgBox.

When you run the macro, the MsgBox action will display a message box that contains a warning or an informational message. The Message argument is the most important argument for the MsgBox action because it determines the text that is displayed in the message box.

4. Click the Message argument box and type Please make sure that you remember to add the employee's phone number!.

There are several other arguments you may want to specify for the MsgBox action, such as the type of icon that is displayed in the message box (None, Critical, Warning?, Warning!, or Information) and the text that is displayed in the title bar of the message box.

5. Click the Type argument box, click the list arrow, and select Information. Click the Title argument box and type Notice.

Your macro should look like the one shown in Figure 10-2. Remember that actions are evaluated and/or executed in the order in which they appear, so the mcrEmployees macro will (1) open the frmEmployees form, (2) maximize the form window, and (3) display the message box.

6. Click the Save button on the toolbar.

That's all we need to do for this lesson.

7. Close the macro window.

8. Double-click the mcrEmployees macro.

Sure enough, the mcrEmployees macro (1) opens the frmEmployees form, (2) maximizes the form window, and (3) displays the message box, as shown in Figure 3.

9. Click OK to close the message box, then click the Close button to close the frmEmployees form.

Working with Macro Groups

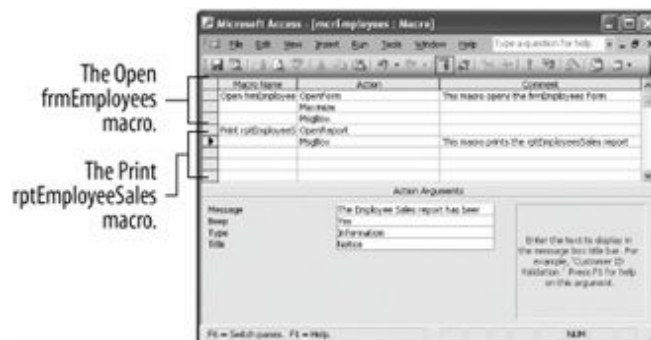


Figure 4. Two macros the Open frmEmployees macro and the Print rptEmployeeSales macro within a single macro group.

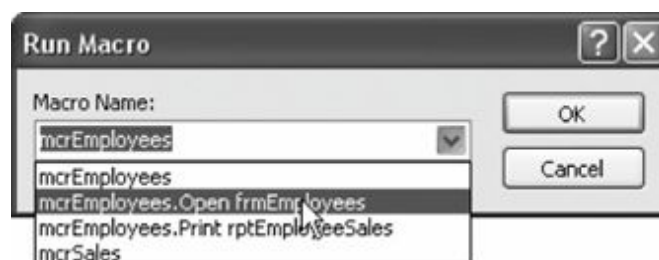


Figure 5. The Run Macro dialog box.

If you are creating lots of macros, you might want to consider organizing them into a macro group to help you manage them. A macro group stores several related macros together in a single macro object. When you create a macro group, you must give each macro in the macro group its own unique name to identify where each macro starts and ends. You do this by entering the macro names in the Macro Name column, which you display by clicking the Macro Names button on the toolbar.

When you combine two or more macros within the same macro group, you must run them separately, by referring to the macro group name, followed by the macro name. For example, mcrEmployees.mcrEmployees frmEmployees refers to the Open frmEmployees macro in the mcrEmployees macro group.

In this lesson you will learn how to group several related macros together in a macro group.

1. Select the mcrEmployees macro and click the Design button.
In order to work with macro, you need to display the Macro Name column.

2. Click the  Macro Names button on the toolbar.

First you need to give the macro you created in the previous two lessons a name. The cursor is already positioned in the Macro Name cell of the first row.

Note: Always enter the macro name in the Macro Name column, next to the Action where the macro starts.

3. Type Open frmEmployees in the first blank cell in the Macro Name column.

The macro name "Open frmEmployees" identifies the macro you created in the previous two lessons. To create another macro in the same macro group, type its name in the Macro Name column next to the first action of the new macro.

4. Press the ↓key three times.

The cursor should be positioned in the Macro Name column next to the first blank Action row. This is where you will add another macro to the macro group.

5. Type Print rptEmployeeSales.

"Print rptEmployeeSales" is the name of the new macro we will create.

6. Click the Action cell to the right of the Print rptEmployeeSales macro name, click the list arrow, scroll down, and select OpenReport.

Similar to the OpenForm action, which opens a form, the OpenReport action opens a report. Next, you need to specify the arguments for the OpenReport action.

7. Click the Report Name text box in the Action Arguments panel, click the list arrow, and select rptEmployeeSales.

This macro will open the rptEmployeeSales report. Notice Print appears in the View argument box this will send the rptEmployeeSales report directly to the printer. Let's add a comment to this new macro.

8. Click the blank Comment box in the Print rptEmployeeSales macro row and type This macro prints the rptEmployeeSales report.

You want to add one more action to the Print rptEmployeeSales macro.

9. Click the Action cell just below the OpenReport action, click the list arrow, scroll down, and select MsgBox.

You need to tell Access what you want the message box to say.

10. Click the Message argument box and type The Employee Sales report has been sent to the printer.

Let's specify several additional arguments for the MsgBox action, such as the type of icon that is displayed in the message box.

11. Click the Type argument box, click the list arrow, and select Information. Click the Title argument box and type Notice.

Your macro should look like the one shown in Figure 4.

12. Click the Save button on the toolbar and close the macro window.

Let's test our new macro. When you combine several macros within the same macro group you must run them separately using the Tools command on the Access menu.

Note: Don't run a macro group by double-clicking it or selecting it and clicking Run. Doing so will run every macro in the macro group often with disastrous results!

13. Select Tools → Macro → Run Macro from the menu.

The Run Macro dialog box appears, as shown in Figure 5. Here's where you select the specific macro you want to run.

14. Click the Macro Name list arrow, select mcrEmployees.Open frmEmployees, and click OK.

Access runs the Open frmEmployees macro.

15. Click OK to close the message box and close the frmEmployees form.

If you want, go ahead and repeat Step 13 and run the Print rptEmployeeSales macro. Make sure your computer is connected to a printer, as this macro will send a copy of the rptEmployeeSales report to the printer.

LAB 7 CREATING CONDITIONAL EXPRESSIONS AND ASSIGNING A MACRO

Lab objective: to study process of creating conditional expressions, assigning a macro to an event, assigning a macro to a keystroke combination

Assigning a macro to an event

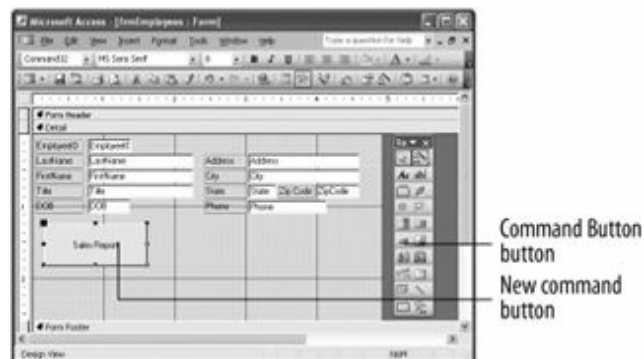


Figure 6. Adding a command button to run a macro.

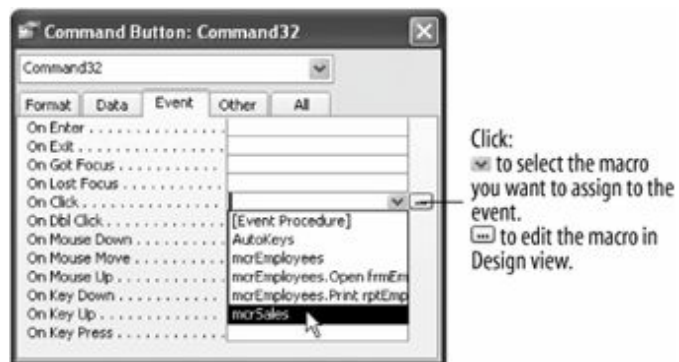




Figure 7. Assigning a macro to a command button's "On Click" event.



Running macros from the Database window or menu is a pain in the neck. That's why most database developers assign macros to controls—particularly, buttons—so that when a user clicks the button or control, a macro is activated.

1. Open the frmEmployees form in Design view.

You want to add a command button to the frmEmployees form to open a report that displays the sales for each employee. First you need to add the command button.

2. Click the  Toolbox button, if necessary, and click the Command Button button on the Toolbox.


The mouse pointer changes to a , indicating that you can click and drag the command button control onto the form.

3. Place the  pointer below the DOB field label and click and drag the  pointer down and to the right to create a command button like the one in Figure 10-6. Click Cancel if the Command Button Wizard appears.

Let's give this button a more meaningful text label.

4. Make sure that the command button is still selected, then click its text label and replace the text with Sales Report. Click anywhere outside of the command button when you're finished.

We're ready to assign a macro to the button to do this you will need to display the command button's Properties.

5. Select the command button, click the  Properties button on the toolbar, and click the Event tab.

The lists all the events to which you can assign a macro most of them you will never use, as you can see in Table 1.

6. Click the On Click box, click the list arrow, and select mcrSales, as shown in Figure 7. Close the Properties dialog box when you're finished.

Let's see how our new command button works.

7. Click the View button on the toolbar to switch to Form view, then click the new Sales Report button.

Microsoft Access runs the mcrSales macro and displays the Employee Commission Report for the current employee. Let's close the report and save our changes...

8. Close the Employee Commission Report and then click the Save button.

Table 1. Event Properties That Can Trigger Macros

Event	Description
Before Update	Macro or function that runs when data in a field or record is changed but before the changes are actually saved to the database. Often used to validate data.
After Update	Macro or function that runs when data in a field or record is changed and is saved to the database.
On Change	Macro or function that runs when the contents of a text box or combo box changes or when you move from one page to another page in a tab control.
On Enter	Macro or function that runs when a control first gets the focus (is selected). The Enter event occurs before the focus moves to a particular control (before the GotFocus event). You can use an Enter macro or event procedure to display instructions when a form or report first opens.
On Exit	Macro or function that runs when a control loses focus (is deselected) on the same form.
On Got Focus	Macro or function that runs when a control gets the focus (is selected).
On Lost Focus	Macro or function that runs when a control loses the focus (is deselected).
On Click	Macro or function that runs when a control is clicked.
On Dbl Click	Macro or function that runs when a control is double-clicked.
On Mouse Down	Macro or function that runs when the user presses the mouse button.
On Mouse Move	Macro or function that runs when the user moves the mouse over a control.
On Mouse Up	Macro or function that runs when the user releases the mouse button.
On Key Down	Macro or function that runs when the user presses a key on the keyboard.
On Key Up	Macro or function that runs when the user releases a key on the keyboard.
On Key Press	Macro or function that runs when the user presses an ANSI key on the keyboard.

Creating Conditional Expressions

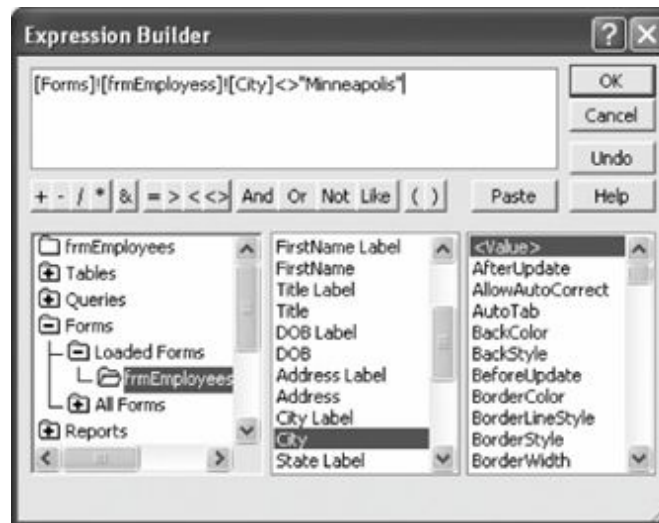


Figure 8. The Expression Builder can help you enter your macros.

[Forms] ! [frmEmployees] ! [City] <> "Minneapolis"
 Database object Form name Control name Evaluation

Figure 9. A conditional expression.

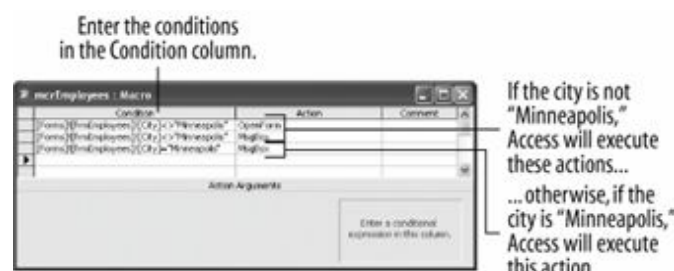


Figure 10. The updated mcrSales macro.


A condition takes action based on a certain condition. For example, if an employee's weekly sales are more than \$2,500, then a condition could calculate a 5-percent commission bonus for the employee; otherwise, it wouldn't calculate a bonus. If you're at all familiar with programming, a condition is similar to an If...Then statement.

You enter conditions in the Condition column in the Macro window. If a condition is true, Access executes the action in that row. If a condition isn't true, Access skips the action in that row and moves to the next row. Conditions often compare values in a specific control on a form or report to a number, date, or constant. For example, the expression in Figure 9 evaluates if the value in a City field is not equal to "Minneapolis." Make sure that you use the proper Microsoft Access syntax when referring to controls in forms or reports.


1. Make sure you have the frmEmployees form from the previous lesson open. Click the View button on the toolbar to switch to Design view.

We want to add a conditional expression to the mcrEmployees macro. If a macro is assigned to a control on a form or report, you can open and edit the macro directly from the form or report without having to access it through the Database window.

2. Select the command button, click the Properties button on the toolbar, click the Event tab, and click the On Click box.

A  Build button appears in every event property. Click this button to create

or modify the macro or Visual Basic procedure assigned to the event.

3. Click the  Build button.

The mcrSales macro appears in Design view.

4. Click the  Conditions button on the toolbar.

The Condition column appears. This is where you need to add the conditions you want Access to evaluate before it executes an action. It's often easier if you use the Expression Builder to help you create your macro conditions.

5. Click the first blank cell in the Condition column and click the Build button on the toolbar.

The appears, as shown in Figure 8.

6. Double-click the Forms folder in the bottom-left window, double-click the All Forms folder, then click the frmEmployees folder.

When you select the frmEmployees folder in the left window, the middle window displays all the controls in the selected form.

7. Scroll down the middle window, and find and double-click the City control.

Access adds Forms![frmEmployees]![City] to the expression area. Now you need to specify how you want to evaluate the City field.

8. Click in the Expression box and add "Minneapolis".

Your expression should look similar to the one in Figure 9.

9. Click OK.

The Expression Builder dialog box closes. The condition you entered will execute the OpenForm action only if the City field is not equal to "Minneapolis." The condition you entered only affects the first row or action in the macro the other actions in the macro will execute without being evaluated. If you want to evaluate the other actions, they must each have their own statement in the Condition column. Let's add some more actions to the macro.

10. Copy the first row in the Condition column and paste it in the second and third rows.

Add another action that will execute only if the City is not equal to "Minneapolis."

11. Click the Action cell in the second row, click the list arrow and select MsgBox. Click the Message argument box and type This is the current commission for non-Minneapolis employees.

Next you need to add an action to perform if the City is equal to "Minneapolis."

12. Edit the expression in the third row of the Condition column so it reads [Forms]![frmEmployees]![City]="Minneapolis".

Your macro should look like Figure 10. Now you need to specify the action to perform if the condition is true.

13. Click the Action cell next to the condition you edited, click the list arrow, and select MsgBox. Click the Message argument box and type Call Linda Ross for the Minneapolis Commission report.

We're finished modifying the macro.

14. Save your changes and close the Macro Design window. Click the View button on the toolbar to display the form in Form view.

Let's test our conditional macro.

15. Find a record whose City field is NOT "Minneapolis" and click the Sales Report button. Click OK and then close the commission report. Click the

Save button and close the frmEmployees form.

Assigning a Macro to a Keystroke Combination

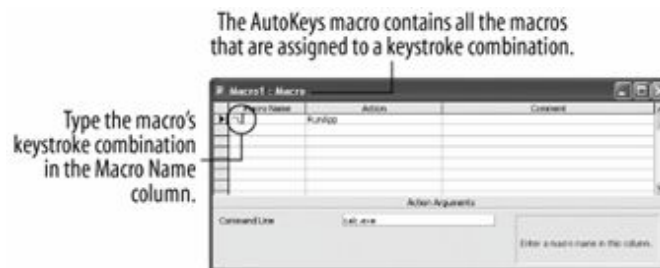


Figure 11. The AutoKeys

Sometimes, instead of assigning a macro to a command button, you may want to assign it to a specific keystroke combination, such as Ctrl + D. Assigning a keystroke combination to a macro makes it fast and easy to access you can execute the macro at any time by pressing its assigned keystroke combination.


Assigning a keystroke combination to macro can be a somewhat complicated process. There are two things you need to know about assigning a macro to a keystroke combination:

- You must create a special macro group, named , which contains all your keystroke-combination macros.
- You type the keystroke combination to which you want to assign the macro in the Macro Name column of the AutoKeys macro window. Enter the keystroke combinations using the examples in Table 2. For example, to assign a macro to the keystroke combination Ctrl + D, you would name the macro ^D.

In this lesson you will learn how to create an AutoKeys macro to assign a macro to a keystroke combination.

1. In the Database window, click the Macros icon in the Objects bar and then click the New button.

The Macro window appears. You need to make sure the Macro Name column is displayed in order to tell Access which keystroke combination you want to assign to the macro.

2. Click the  Macro Names button on the toolbar, if necessary.

Now you need to type the keystroke combination to which you want to assign the macro. Table 10-2 shows the key combinations you can use to make key assignments in an AutoKeys macro group. We want to assign a macro to the keystroke combination Ctrl + L. Here's what you need to enter:

3. Type ^L in the first blank Macro Name cell, as shown in Figure 11.

The name ^L refers to Ctrl + L. The ^ (caret) signifies the Ctrl key and the "L" signifies the "L" key.

Note: If you assign a macro to a keystroke combination that Microsoft Access is already using (for example, Ctrl + X is the keystroke combination for the Cut command), the macro you assign to this keystroke combination will override the Microsoft Access keystroke combination assignment.

Now we need to specify what we want the macro to do.

4. Click the first blank Action cell, click the list arrow, and select RunApp from the list.

The RunApp action starts another program, such as Microsoft Excel or Word. We want the RunApp action to start the Calculator program. You need to specify the name and location of the program you want to run in the Action Arguments area.

5. Click the Command Line text box and type calc.exe.

Now let's save the macro, making sure that we name it AutoKeys.

6. Click the Save button on the toolbar, type AutoKeys, and click OK.

We're ready to test our AutoKeys macro.

7. Press Ctrl + L.

Microsoft Access executes the macro assigned to the Ctrl + L keystroke combination and starts the Calculator application.

8. Close the AutoKeys macro and database.

Give yourself a pat on the back if you've gotten through this tutorial. You've just learned how to automate your Microsoft Access databases to work more like a full-featured application instead of a dull, static database.

Table 2. The SendKey Syntax

Heading	Heading
Ctrl + Any Key	^ (For example, enter ^E for Ctrl + E.)
Shift + Any Key	+ (For example, enter +E for Shift + E.)
Alt	% (For example, enter %E for Alt + E.)
Enter	{ENTER}
Esc	{ESC}
Tab	{TAB}
Insert, Delete	{INSERT} or {INS}, {DELETE} or {DEL}
Page Down, Page Up	{PGDN}, {PGUP}
Home, End	{HOME}, {END}
Arrow Keys	{UP}, {DOWN}, {LEFT}, {RIGHT}
Caps Lock	{CAPSLOCK}
Function Keys	{F1}, {F2}, {F3}, etc...

Summary

Creating and Running a Macro

- To Create a Macro: In the database window, click the Macros icon in the Objects bar and click the New button. Click the first blank Action cell, click the list arrow, and select the action you want the macro to perform. Specify any required arguments for the action in the Action Arguments area. Repeat for each additional action you want the macro to execute. Click the Save button on the toolbar, give your new macro a name, and click OK.

- To Run a Macro: Click the Macros icon in the Objects bar and double-click the macro you want to run.

Editing a Macro

- To Modify a Macro: In the Database window, click the Macros icon in the

Objects bar, select the macro you want to edit, and click the Design button.

Working with Macro Groups

- To Create a Macro Group: Create a new macro or edit an existing macro, then click the Macro Names button on the toolbar. Type the macro name in the Macro Name column next to the action where the macro starts. If necessary, add the macro actions or edit the existing macro actions. Save the macro and close the macro window.
- To Run a Macro in a Macro Group: Select Tools → Macro → Run Macro from the menu, click the Macro Name list arrow, select the macro you want to run, and click OK.

Assigning a Macro to an Event

- To Assign a Macro to a Control on a Form or Report: Open the form or report in Design view, click the control to which you want to assign the macro and click the Properties button on the toolbar. Click the Event tab, click in the box for the type of event you want to assign to the macro, then click the list arrow and select the macro you want to assign to the event. Close the Properties dialog box and save the form or report.

Creating Conditional Expressions

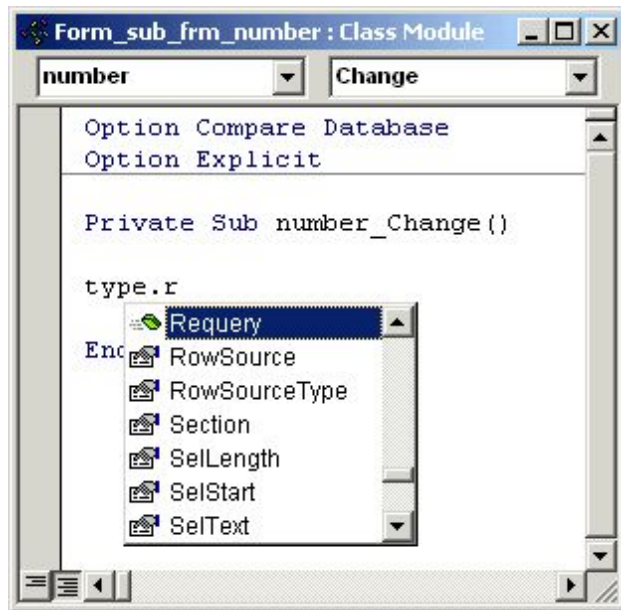
- To Create a Conditional Expression in a Macro: Create a new macro or edit an existing macro. In Design view, click the Conditions button on the toolbar and click the Condition cell next to the action you want to evaluate. Enter the conditional expression in the Condition cell, using proper Access syntax. You can use the Expression Builder to help you create the expression by clicking the Build button on the toolbar. Repeat these steps for each action you want to evaluate. Save the macro and close the macro window.

Assigning a Macro to a Keystroke Combination

- To Assign a Macro to a Keystroke Combination: Create a macro group named AutoKeysthis will store all the macros that are assigned to a keystroke combination. Click the Macro Names button on the toolbar. Type the keystroke combination in SendKey syntax in the Macro Name column next to the Action where the macro starts.

LAB 8 CREATING MODULE

Lab objective: to study process of creating module



Most of the things that your system needs to do can be accomplished using standard Access functions. The sorts of things you might need Visual BASIC for Applications (VBA) for are:

- Requerying subforms and combo-boxes (discussed above)
- Enabling, disabling and hiding controls (discussed in the Forms section)
- Checking for overlapping appointments (discussed in the Practical Examples section)
- Displaying confirmation messages using message boxes
- Copying values from one field or form to another

VBA functions are linked to events in your system, such as buttons being clicked, fields being changed, or even forms being opened and closed. To enter the code, go to the *Properties* for your chosen object and click the *Event* tab. In there will be a list of all of the events for the object (not all objects have the same events – forms will have an *OnResize* event, for example, but buttons won't, because they aren't resized in normal use). Click on your chosen even, and the ellipsis (...) button will appear to the right. Click that button and choose *Code Builder*.

Message Boxes

A useful feature of VBA is that you can display pop-up messages to convey information to the user, and ask for confirmation for certain actions, such as closing the system. These are known as *message boxes* and there is a command in VBA called *msgbox()* that allows you to control their appearance.

If you just want to display a message (with an OK button), the syntax is quite simple. For example, to display the word *Hello* in a message box, the command would be:

```
msgbox("Hello")
```

Note that any text you want to display must be enclosed in speech marks.

There are more options that you can select, but be aware that if you enter more than just the message, then the *msgbox()* command returns a value, so you use a variable or another command, such as *if* to handle the value it returns.

As you type a command, the VBA editor will help you with the options, displaying drop-down lists of all the options at each point – the only thing you really need to remember is that all of the options are separated by commas. You can also use the help, of course, by highlighting your command and pressing the F1 key.

If you want to display a message box, with the title “Confirmation”, the message “Are you sure?” and Yes and No buttons, for example, then you would use the following code:

```
msgbox("Are you sure?",vbYesNo,"Confirm")
```

Remember that all text values need to be in speech marks. Remember that when you use *msgbox()* in this way, it returns a value, so you can't use the command on its own – probably the best thing to do would be to use *if*, so if you wanted to add some confirmation messages to your system, you could do something like this (on the OnClick event of your Exit button):

```
If MsgBox("Are you sure you want to exit?", vbYesNo, "Confirm Exit") = vbYes Then
    Application.Quit
```

(this is in a small font so that it all fits on one line – lines in the VBA editor are much longer).

The If Command

The *If* command is used to make decisions in your macro, and can be used either on one line, or as a more complex version with an *else* clause. The general format is either:

```
If test_condition then action
```

Or, if you want to have an alternative action:

```
If test_condition then
```

```
    action
```

```
Else
```

other action

End if

Setting the Values of Fields

You can set the value of a particular field (or, indeed, any other property, such as visibility, or whether the field is enabled) from your code quite simply by accessing the properties of them using the name that is at the top of the *All* tab in the properties. For example:

```

field_name.value = "Hello"
field_name.value = date()
field_name.value = other_field.value
field_name.enabled = TRUE
field_name.visible = FALSE

```

Remember that you need to enclose all text values in speech marks. You can also use functions such as *date()*, and the predefined constants *TRUE* and *FALSE* (which do not need to be in speech marks).

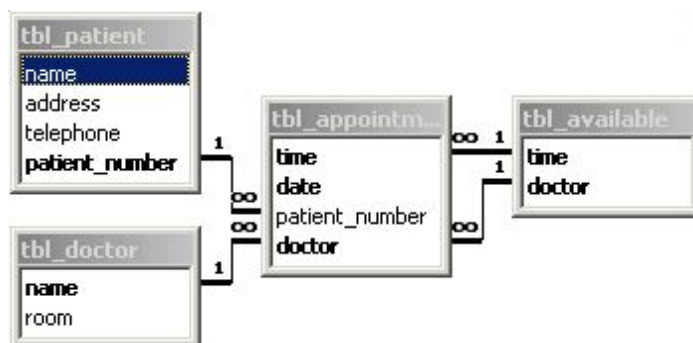
Practical Examples

This section of the booklet demonstrates some approaches you could take to common problems in Access. It goes without saying that you should not copy the databases or any of the code without crediting it in your project.

Clashing/Overlapping Appointments

A common type of A level project is a booking system created in Access. One of the most important functions such a system will perform is to prevent double-bookings or clashes. This can be more or less tricky, depending on how the appointments are made. Appointments or bookings fall into two basic categories - either discrete appointments (e.g. a doctor's surgery, where a patient is given a 10-minute slot), or bookings with a start and end time/date (e.g. a booking in a holiday cottage, with arrival and departure dates).

If you're booking hour lessons, or discrete appointments of some sort, then you can use referential integrity to stop double-bookings. If you're booking something like holiday cottages, on the other hand, where there are arrival or departure dates and the bookings can overlap, then it's a little more tricky. On this page I shall propose the methods I would use to prevent double-bookings in each case.



Discrete Appointments

In the diagram below you can see the tables and relationships from appointments.mdb (available in Shared Document). It's the classic three-table booking system, with

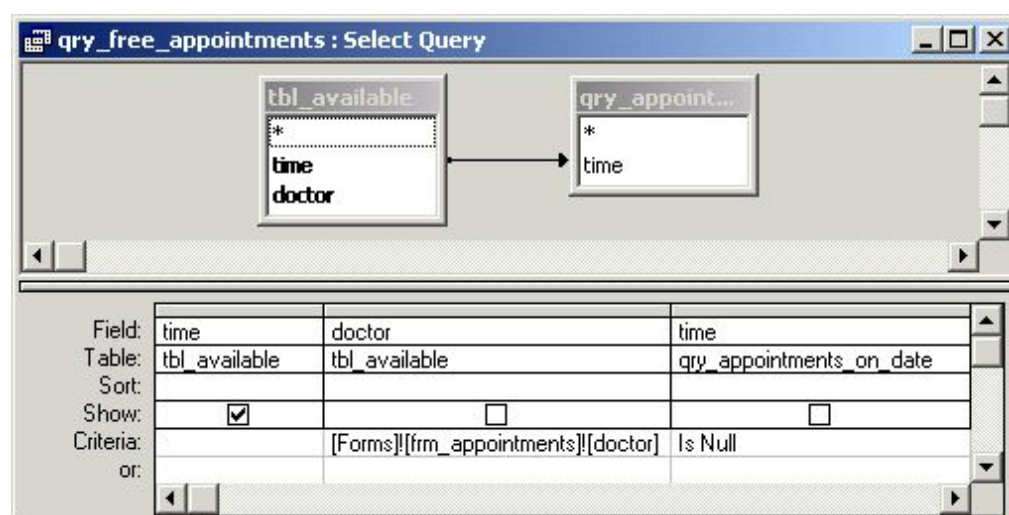
patients, doctors, and a linking appointments table. I've also added a fourth table (*tbl_available*) that contains a list of all the available appointment slots for each doctor. To make things easier, in this example the surgery has the same appointment times each day, although you could also add the day of the week if you wanted to vary their times.

Referential integrity will prevent the receptionist from making a booking for a doctor in a time slot that doesn't exist in *tbl_available*, but what happens if the appointment is already taken *on that particular day*?

One of the tidiest ways to prevent double-bookings is to use a combo-box to present the user with only the available appointments for a given day (see the form on the left). This can be populated from a query, and requeryed whenever the date and/or doctor are changed.

Although the idea of the query - to find available slots on a given day - sounds quite simple, it's not as easy as it first appears. This is because you're looking not just for bookings that don't exist, but bookings that don't exist on a particular day.

My solution, therefore, uses two queries - one that finds the appointments that have been made for a particular doctor on a given day (straightforward enough), and then a second one that effectively subtracts these from the complete list of available slots (i.e. those in *tbl_available*), to give the available appointments for the chosen day.



The illustration to the right shows this second query. Note that it contains a table and a

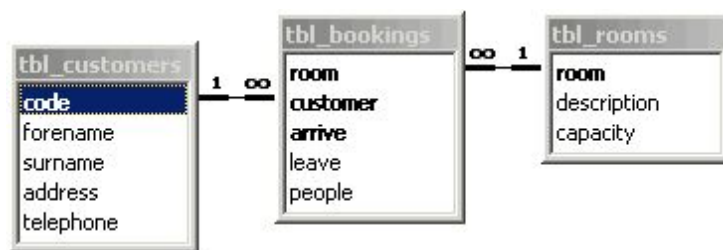
query - you will have to create the relationship yourself (you do this by dragging the time field from the table to the query, as you would if you were joining two tables). You also need to change the *Join Properties* (right click on the line joining the table and the query), to select the middle option - i.e. to show all the records from *tbl_available...*

Once you've done that, the query itself is quite simple - you just want all the times for your doctor from *tbl_available* where there isn't a corresponding time in *qry_appointments_on_day* - i.e. where the time is NULL.

If you now use the form to make a booking for a particular doctor on a given day, you will notice that the time disappears from the list once you've clicked Save, so that that slot can't be used again.

Variable-Length Bookings

Preventing clashes with variable-length bookings is a little more involved. If you've trawled the newsgroups and Access FAQs, then you may have come across the *cartesian product* method for picking up double-bookings. The problem with this approach is that you do the checking after the record has been saved, by which time it is too late.



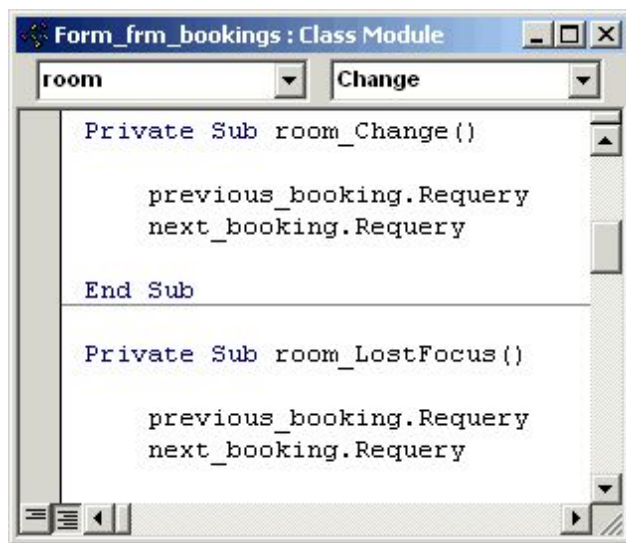
In *clash_test.mdb* (in Shared Documents) I've proposed a method for preventing the user from saving clashing bookings in the first place. It isn't overly elegant, and you may need to tweak the events on which the query is re-run to suit your needs, but hopefully it'll give you some idea of how to approach a solution.

What happens is that the form used to enter the date has two hidden fields containing the dates of the latest prior booking, and the first subsequent booking, in the same room. When the *Save Record* button is clicked, a macro compares the dates and stops the save if the dates overlap.

The database itself is just a simple three-table affair, with tables for rooms, customers, and bookings. Generally, things like customer codes are frowned upon, but I've just used

them here for simplicity. I've added fields for the number of guests in the party, but I haven't actually done anything with them - the point of the database is only to demonstrate the double-booking validation.

On top of the bookings table, I've created a form (shown to the left) that contains the fields in the table. I've used combo boxes for the customer and room number to make it easier to select rooms and customers that exist and maintain referential integrity.



Underneath the *Save Record* and *Cancel* buttons are two invisible combo-boxes that are populated from the two queries, *qry_previous_booking* and *qry_next_booking*. The *visible* property of both combo-boxes is set to *False* so that they are hidden.

The query *qry_previous_booking* finds the latest prior booking in the same room, based on the dates and room you enter. Because you might change the dates or the room number, you need to put a *.requery* command on the *LostFocus* or *Change* events to re-run the query. All that does is ensures that the next and previous bookings always relate to the rooms and dates you have entered.

The next step in the validation process is to check that the appointments don't overlap. This needs to be done before the record is saved, so that you can warn the user and request alternative dates or a different room before it's too late. The easiest way to do this is to add the *Save* button using the normal wizard, and then add some VBA code afterwards.


```

Form_frm_bookings:Class Module
save_button Click
Private Sub save_button_Click()
On Error GoTo Err_save_button_Click

Dim before, after As Integer

If previous_booking.ItemData(0) <> "" Then
    before = DateDiff("d", previous_booking.ItemData(0), arrive)
End If

If next_booking.ItemData(0) <> "" Then
    after = DateDiff("d", leave, next_booking.ItemData(0))
End If

If before >= 0 And after >= 0 Then
    DoCmd.GoToRecord , , acNewRec
Else
    If before < 0 Then
        MsgBox ("There will still be someone in the room on this date!")
    End If
    If after < 0 Then
        MsgBox ("Someone is due to arrive before this departure date!")
    End If
End If
End Sub
Err_save_button_Click:
End Sub

```

The code just uses a simple *if... then... else...* loop to display a message if the appointment clashes with another, or save the record if it doesn't. To determine whether the start of the new booking comes before the end of the previous one, or whether the end of the new booking comes after the start of the next one, I've used the *DateDiff()* function. This will give either a positive or a negative answer, depending on which date comes first. You might find it simpler to try a simple comparison, but I've found that *>* and *<* don't always work reliably with dates.

The code for the *Save* button is shown right. Note that you can put most of the *if* statements on one line, and omit the *End Ifs* - I've just done it that way so that the code isn't too wide to fit on the screen here. One last thing to bear in mind is that the record will still be saved if you click on the button at the top of the window to close the form. You can get around this either by disabling the Close button (in Form properties), or by adding similar code to the form's Close event to check for clashes there too.

TASK VARIANTS

1. Private Banking Operational Data
2. Investment Banking Operational Data
3. Retail Banking Operational Data
4. Combined Investment and Retail Banking Operational Data
5. Debt collection.
6. Debt management.
7. Online banking.
8. Credit Card Online.
9. Mobile Banking.
10. e-Monies Electronic Fund Transfer.
11. Online Payment of Excise & Service Tax (9).
12. Phone Banking.
13. Bill Payment.
14. Smart Money Order.
15. Card to Card Funds Transfer.
16. Funds Transfer (e-Cheques).
17. Anywhere Banking.
18. Internet Banking.
19. Mobile Banking.
20. Bank@Home.

APPENDIX A
MACRO ACTION REFERENCE

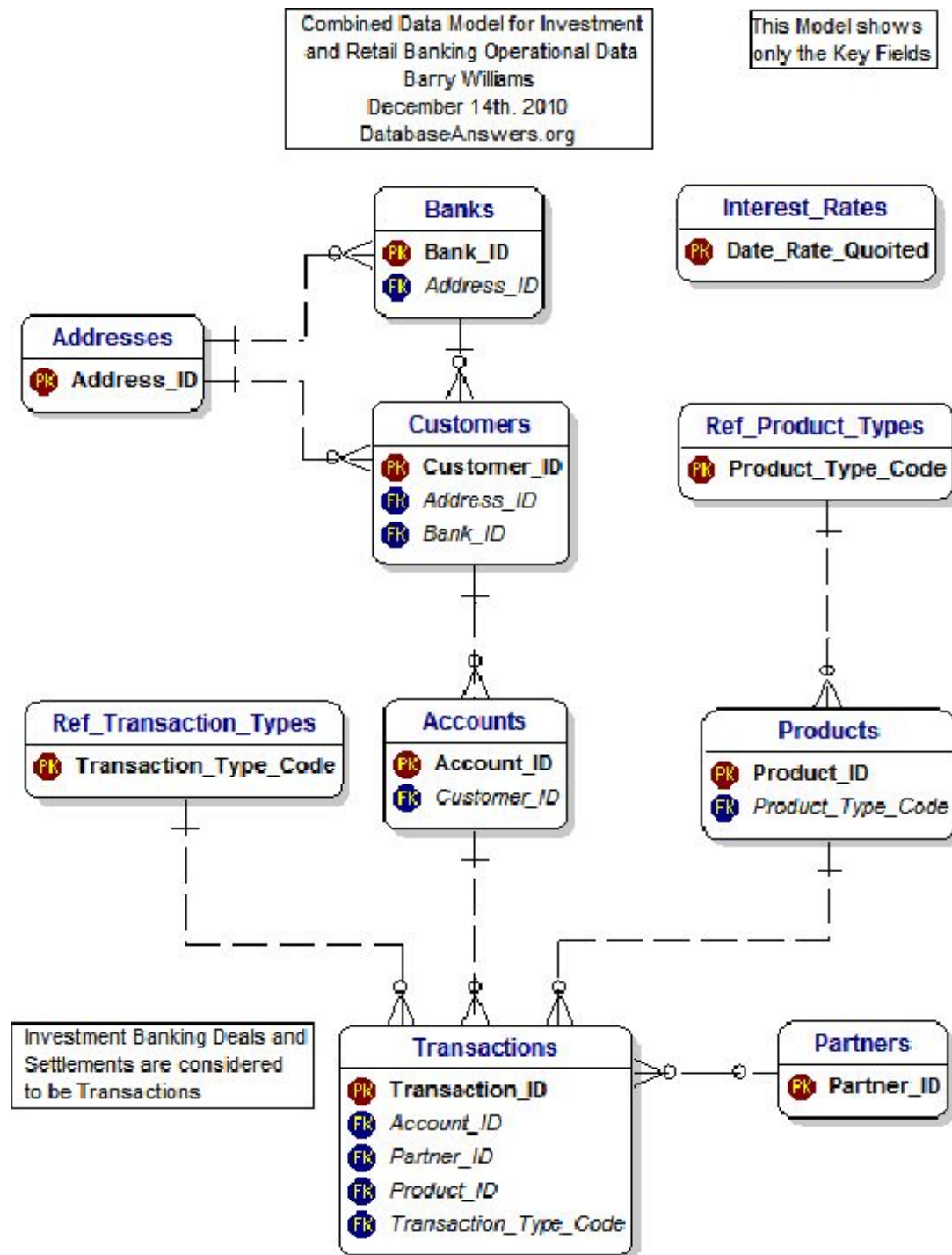
Action	Description
AddMenu	Adds a menu to a custom menu bar for a form or report. Each menu on the menu bar requires a separate AddMenu action.
ApplyFilter	Applies a filter or query to a table, form, or report.
Beep	Causes the computer to beep.
CancelEvent	Cancels the event that caused the macro to run.
Close	Closes the specified window or the active window if none is specified.
CopyObject	Copies the specified database object to a different Microsoft Access database or to the same database with a new name.
DeleteObject	Deletes the specified object or the object selected in the Database window if no object is specified.
Echo	Hides or shows the results of a macro while it runs.
FindNext	Finds the next record that meets the criteria specified with the most recent FindRecord action or Find dialog box. Use to move successively through records that meet the same criteria.
FindRecord	Finds the first or next record that meets the specified criteria. Records can be found in the active form or datasheet.
GoToControl	Selects the specified field on the active datasheet or form.
GoToPage	Selects the first control on the specified page of the active form.
GoToRecord	Makes the specified record the current record in a table, form, or query. Use to move to the first, last, next, or previous record.
Hourglass	Changes the mouse pointer to an hourglass while the macro runs.
Maximize	Maximizes the active window.
Minimize	Minimizes the active window.
MoveSize	Moves and/or changes the size of the active window.
MsgBox	Displays a message box containing a warning or informational

Action	Description
	message.
OpenForm	Opens a form in Form view, Design view, Print Preview, or Datasheet view.
OpenModule	Opens the specified Visual Basic module in Design view.
OpenQuery	Opens a query in Datasheet view, Design view, or Print Preview.
OpenReport	Opens a report in Design view or Print Preview or prints the report immediately.
OpenTable	Opens a table in Datasheet view, Design view, or Print Preview.
OutputTo	Exports the specified database object to a Microsoft Excel file (.xls), rich-text file (.rtf), text file (.txt), or HTML file (.htm).
PrintOut	Prints the active database object. You can print datasheets, reports, forms, and modules.
Quit	Quits Microsoft Access.
Rename	Renames the specified object.
RepaintObject	Completes any pending screen updates or pending recalculations of controls on the specified object or on the active object if none is specified.
Requery	Forces a requery of a specific control on the active database object.
Restore	Restores a maximized or minimized window to its previous size.
RunApp	Starts another program, such as Microsoft Excel or Word.
RunCode	Runs a Visual Basic Function procedure.
RunCommand	Runs a command from Microsoft Access's menus for example, File → Save.
RunMacro	Runs a macro.
RunSQL	Runs the specified SQL statement for an action query.
Save	Saves the specified object or the active object if none is specified.
SelectObject	Selects a specified database object. You can then run an action that applies to that object.
SendKeys	Sends keystrokes to Microsoft Access or another active application.

Action	Description
	These keystrokes are processed as if you had typed them yourself on the keyboard.
SendObject	Sends the specified database objects as an attachment in an e-mail.
SetMenuItem	Sets the state of menu items (enabled or disabled, checked or unchecked) on custom menus. Works only on custom menus created using menu bar macros.
SetValue	Sets the value for a control, field, or property on a form or report.
SetWarnings	Turns all system messages on or off. This has the same effect as clicking OK or Yes in each message box.
ShowAllRecords	Removes any applied filter from the active table, query, or form.
ShowToolbar	Shows or hides a built-in toolbar or a custom toolbar.
StopAllMacros	Stops all currently running macros.
StopMacro	Stops the currently running macro. Use to stop a macro when a certain condition is met.
TransferDatabase	Imports or exports data to or from the current database from or to another database.
TransferSpreadsheet	Imports data from a spreadsheet file into the current database or exports data from the current database into a spreadsheet file.
TransferText	Imports data from a text file into the current database or exports data from the current database into a text file.

APPENDIX B

DATA MODEL FOR COMBINED INVESTMENT AND RETAIL BANKING OPERATIONAL DATA



LITERATURE

1. Andersen V. How to do everything with Microsoft Office Access 2007. – NY: McGraw Hill, 2007. – 617 p.
2. Baker K., Powell G. Understanding Financial Management: A Practical Guide. – NY: Wiley-Blackwell, 2005. – 504 p.
3. Benson S. Information Systems: A Business Approach. – NY: Wiley, 2008. – 400 p.
4. Mishkin F. S. The economics of money, banking, and financial markets. – NY: Addison-Wesley, 2004. – 850 p.
5. Saunders C.S., Pearlson K.E. Managing and Using Information Systems. – NY: Wiley, 2009. – 400 p.
6. Carroll M.L. CyberStrategies: How to Build an Internet-Based Information System. - NY: Wiley, 1995. – 286 p.
7. Matthews K., Thompson J. The economics of banking. – NY: John Wiley & Sons Ltd, 2005. – 257 p.
8. Thomsen E. OLAP Solutions: Building Multidimensional Information Systems. - NY: Wiley, 1997. – 608 p.