

Database Case 6

Mak Audio 1

Problem: Develop a mailing list database

Management skills: Organize
Control

Access skills: Database Design
Queries
Reports
Linking Data Tables

Data Tables: MAK_A
MAK_B

Mak Audio is a medium-sized company based in Nashville, Tennessee. The business was established in the early 1970s by Allan and Rick Mak, and grew out of a hobby the two brothers shared in music and electronic circuitry. The company has two distinct business divisions. The Mak Design side of the business designs, manufactures and retails a range of audio equipment at the high end of the market. This includes custom built sound systems, the famous Mak valve amplifier which is eagerly sought by audiophile buffs around the world, and high performance ribbon speakers. This side of the business is high margin and very successful, and is managed by Allan Mak.

Mak Design's main customers are audio retailers, and music production houses such as EMI, Warner Music, and Polygram Records. Every six months, Mak Design distributes a product catalogue to its 3,500 customers.

Rick Mak manages Anabasis, the other division in the company. Anabasis is a lucrative mail order business which locates and purchases rare and hard-to-find sound recordings through contact agents around the world. Many of these items are sourced from the deceased estates of musical performers. Music areas include classical, opera, jazz, rock, country and western, folk, soul, reggae, and ethnic (Asian and African). The business currently carries around 15,000 titles in this area, which are held in audio CD, vinyl record, tape or video format. These are sold to audio retailers and music production houses throughout the world. Currently Anabasis has around 12,000 customers on its mailing list, and like Mak Design, distributes a catalogue of new and current offerings on a biannual basis.

The existing Anabasis mailing list presents a difficulty for Rick Mak. While the marketing side of Mak Design has been handled by a PC-based database package for some years, Mak has been unsure how to go about integrating the Anabasis mailing list into this system, although he knows this could make good business sense. Mak suspects that at least 25% of the customers on his mailing list are also customers who regularly purchase from Mak Design.

The Anabasis mailing list has been steadily growing since the mid 1970's, when this side of the business was formed. Originally the mailing list was kept on a series of cards which recorded customer mailing and music interest details. This method soon became unwieldy as the list grew. In the late 1970's, Anabasis outsourced the data management of its list to Good Code Computer Services. Good Code provides data entry and maintenance services, and produces mailing label printouts for Anabasis whenever a catalogue mailout is required.

Mak has a number of problems with the bureau method. He has recently received an invoice from Good Code requesting its usual quarterly payment of \$2,500 for maintenance of his mailing list. Mak has been concerned by the escalating costs of using Good Code for some time and is no longer sure their services are worthwhile. He also feels that the bureau method does not give him control of his data.

The current mailing list is not a database. In the 1970s, Good Code created the original list as a long sequential file organised alphabetically by customer last name, and has continued to maintain it in this format. Mak is not happy with this structure which he feels is inflexible, and cannot provide information vital to the management of his business. For example, he cannot easily obtain information about the music interests of his customers for targeted mailouts. A simple query like *"print a list of all Australian customers with an interest in Funk music"*, requires a special programming task by Good Code, a lengthy delay before Mak receives the information, and incurs an additional charge for Anabasis. More complex queries are simply impossible.

The structure of the current mailing list also makes it difficult to update existing data or to check whether a customer is already on the list. Mak knows that a large proportion of his mailing list is probably out of date. He has noticed that the number of return-to-sender envelopes Anabasis receives has rapidly increased as customers move addresses or contact names change. The present system makes it difficult to incorporate these changes. In the meantime, Anabasis is incurring a lot of unnecessary mailout associated expense. Mak has also noticed that some of these returns have been sent to the same customer several times. This means that there is a level of data duplication in the mailing list. From current indications, Mak suspects it could be as high as 20%.

Another inherent problem with the current system is that if there are 45 customers at a large music company like Warner Music, the name and address of the company is repeated 45 times! If the name of the company were to change (as is common in the music business) the new name would need re-entered 45 times!

Mak is frustrated by the deficiencies of his current mailing list. Although Anabasis is profitable, Mak suspects it could be more so if he had the right information when he wanted it, and was able to do targeted mailouts with ease. He has decided that the only way to get what he wants is to build a parallel mailing list system from the ground up. After consultation with his brother, Mak has purchased a Pentium-based microcomputer, and the same relational data base package used by Mak Design. Once the new mailing system is working satisfactorily, he plans to dispense with Good Code's services. At a later stage, he will consider integrating his system with that of Mak Design.

The existing mailing system has the right information, but it is not stored efficiently, and it is not easy to interrogate. The existing system has the customer's name, primary music interest (MusIntA), secondary music interest (MusIntB), organisation name, address, city, state, country and zip code. The music interests of customers are coded using an internally developed schema.

To make storage and updating of the mailing list more efficient, Mak wants the data stored in two tables with one containing the names of the customers, and the other the name and addressing details of the company they work for. Whenever a mailing list is required, the two tables could be joined and the results printed. Mak needs your assistance in developing this project.

The Access data tables on your *SolveIt!* diskette MAK_A and MAK_B provide the overall design and sample data for the new mailing system. These tables form the basis of this case.

Tasks: There are six tasks in this case:

1. Link the two tables MAK_A and MAK_B on the Org field. It may be helpful to print out the structure of these tables first. Then print out the results of this join.
2. Create a report for marketing purposes that lists all customers separated into alphabetical organisational groups. The fields describing reading interests and address are not needed. Each group should contain one organization with the records sorted alphabetically by last name within these groups.

Design this report as a professional document. Include a title, group headings, and any other useful features you think of.

3. Print the report.
4. Design a custom label with four lines of information that can be used as a mailing label.
5. Anabasis has recently obtained some rare Soul and Jazz sound recordings. Print a report of all USA customers with one of two music interests (i.e. 64 or 77) in these areas, as either their primary or secondary music interest. Be careful with the Boolean logic in this problem.
- *6. The current system uses codes (e.g. 64, 77) for data in the music interest fields. The user of the system and recipients of its reports must know what these codes mean. Devise a way to include the meaning for these codes into the database. You will need to make up meanings for each of the music interest codes (e.g. 64 is Jazz Music and 77 is Soul Music).

Make sure that the method you devise does not include data redundancy. It would be poor database design to have the meanings for these codes repeated (redundantly) throughout the database. From your new table structure, produce a report of customers, their organisation and music interests which does not include codes.

Time Estimates (excluding task marked with *):

Expert: 45 minutes
 Intermediate: 1.5 hours
 Novice: 3 hours

Tutorial For Database Case 6 Using Access 97

In this case, you will learn two important new skills: joining table files (creating relationships), and creating labels in reports.

Joining Tables

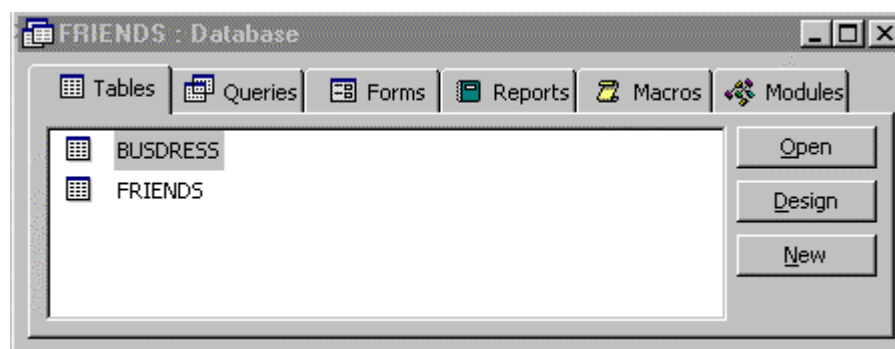
In many earlier database applications, the joining of tables is a procedure that would require you to generate programming code. Access allows you to do this simply via the menu or toolbar, and to run the results of the join in a query (or a report). Access will also generate the SQL programming code for the procedure, so that you can see the effect of your actions.

Before you can join two tables, the related field must be present in both tables. While the field names do not have to be the same, for the join to work, the two fields must contain matching data within records. For this procedure, we will be using the FRIENDS database, and a second practice table BUSDRESS. The latter contains the last names and business addresses of persons in the FRIENDS database.

1. Load the practice database FRIENDS.MDB. Click on the *Table* object and then import the second practice database file BUSDRESS.DBF using FILE/GET EXTERNAL DATA and then IMPORT from the menu or by clicking on the *Import* toolbar button. Recap the tutorial in Chapter 4, if you are unsure about importing external files. BUSDRESS is a table contained within the SOLVEIT master database, so make sure that Access is selected as the format during the import procedure. FRIENDS.MDB should now have two tables, and look similar to Figure 5-65.



Figure 5-65



2. From the Database Window, highlight and open the FRIENDS table, and then repeat this procedure for the BUSDRESS table. Compare the two tables, noting that both LastName fields contain identical data. Remember that the field names themselves do not have to be the same. Close the tables in turn by double clicking on the box in the top left hand corner of each.

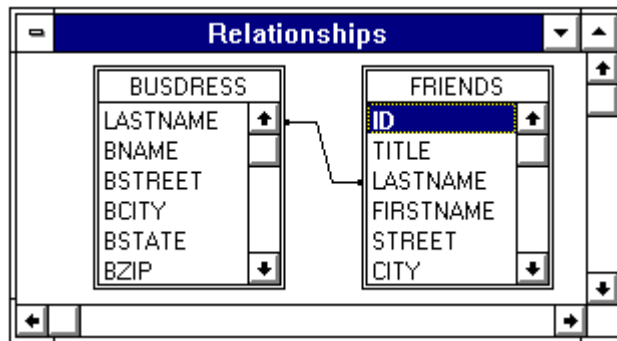
3. From the Database Window, create a relationship between the two tables. This will speed up the execution of the query we will create in a moment. Table relationships were last discussed in the tutorial for Chapter 4.



4. To create a relationship: click the *Relationships* button on the toolbar. From the Show Table dialog box, highlight the BUSDRESS table and click Add. Repeat this action for the FRIENDS table.

5. Highlight the LastName field in the BUSDRESS table and while holding down your left mouse button, drag this over to the LastName field in the FRIENDS table. Release the mouse button. A Relationships dialog box will appear. Now click on the *Create* button. Access generates a dynamic link (line) between the two known as an *equi-join*. An equi-join selects all the records from both tables that have the same value in the field selected as being common to both. Your Relationships window should look similar to Figure 5-66.

Figure 5-66



6. Save the relationship by choosing FILE/SAVE or clicking on the *Save* toolbar button.


7. Press F11 to return to the Database Window.

You are now ready to create a query to generate the effect of the joining action.

8. From the Database Window, click on the *Query* object, and click


New. Now either select the *Simple Query Wizard* or the *Design View* option from the New Query dialog box. This tutorial assumes that you have selected the *Design View* option.

9. From the *Add Table*, highlight the BUSDRESS table and click Add. Repeat this action for the FRIENDS table. Click Close. Notice that Access has remembered the table relationship, and has immediately established a join between the two LastName fields.

10. Select VIEW/SQL VIEW from the menu or click on the SQL toolbar button  to see the equivalent of this action in the Access Basic programming code:

```
SELECT
FROM BUSDRESS INNER JOIN FRIENDS ON BUSDRESS.LAST_NAME = FRIENDS.LAST_NAME;
```



Click on the design view button  to return to the Query Design window or select VIEW/DESIGN VIEW from the menu.

11. From the query window, select and drag the following fields down into the Field bar of the QBE Grid.

From FRIENDS

Title
 FirstName
 LastName

From BUSDRESS

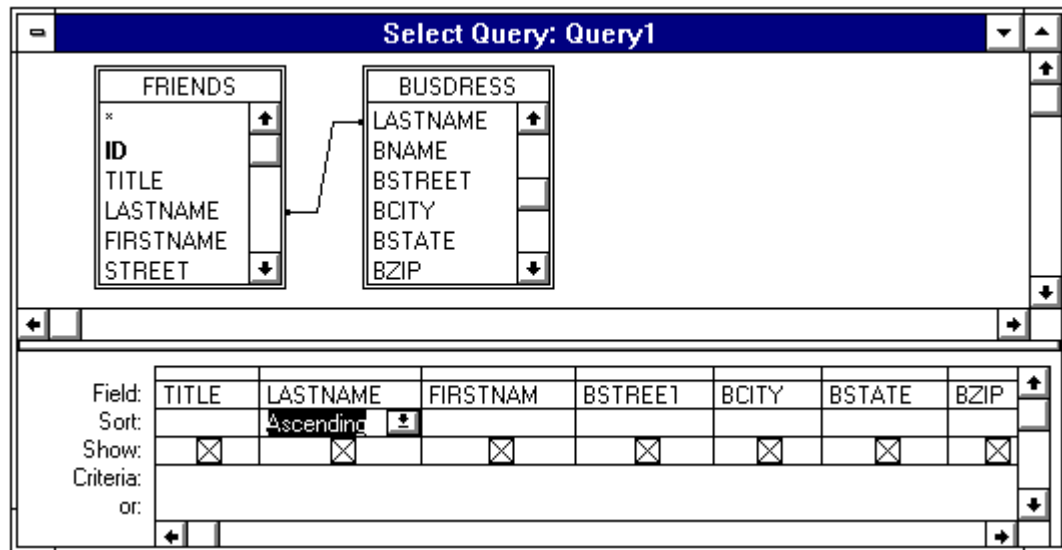
BStreet
 BCity
 BState
 BZip


Lastly, click on the *Sort*: bar in the LastName field, and choose Ascending order. Your query window should now look similar to Figure 5-67.

12. Click on the SQL button again, to see the effect this action has had on the Access Basic programming code:

```
SELECT FRIENDS.TITLE, FRIENDS.LAST_NAME, FRIENDS.FIRST_NAME, BUSDRESS.BSTREET,
BUSDRESS.BCITY, BUSDRESS.BSTATE, BUSDRESS.BZIP
FROM BUSDRESS INNER JOIN FRIENDS ON BUSDRESS.LAST_NAME = FRIENDS.LAST_NAME
ORDER BY FRIENDS.LAST_NAME;
```

Figure 5-67



13. Click on the *Run*  query toolbar button, and view the resulting dynaset. Access has matched and merged the business address data in the BUSDRESS table with the name data in the FRIENDS table to create a composite display containing non-duplicative data from both (see Figure 5-xx).

14. Print the query if you wish to, or use as the basis for a new report.

15. Save the query (eg: *Tute6 Query1*), and press F11 to return to the Database Window.

Figure 5-68

FRIENDS

Joined on

BUSDRESS

Query: Query1						
	TITLE	LASTNAME	FIRSTNAME	BSTREET	BCITY	BSTATE
▶	Dr	Drucker	Peter H.	345 Ohio St.	Cleveland	Ohio
	Mr	Whitney	Craig	345 Westin	Portland	Oregon
	Mr	Sitkin	Howard W.	4523 8th Avenue	Hartford	Conn
	Dr	Skalek	William F.	57 Morris	Seattle	Washington
	Prof	Salione	Phillip	459 Palm Drive	Springvale	California
	Mr	Fabian	James T.	98 Kansas Street	Burlington	Vermont
	Mr	Kohlman	Frank	856 Irvine	Irvine	California
	Mr	Tedesco	George R.	98 Lakes Drive	St. Paul	Minnesota
	Ms	Zito	Helen K.	23 Ocean Drive	Cambridge	Mass.
	Dr	Peterson	Jack S.	900 Lake Drive	Detroit	Michigan
	Dr	Nelson	Robert M.	75 Yorktown Circle	Yorktown	New York

Record: 1

of 11

◀


▶

◀▶


◀▶

How to Create Labels in Access


A report designed to print names and addresses on labels is a common feature of most database applications. The report feature in Access includes a powerful Mailing Label wizard, with sizes to match most common labels. Let's use the sample FRIENDS database to create a simple three line address label.

1. Load FRIENDS.MDB. Click on the Report object, and click New. From the New Reports box, click on the  button to the right of the *Choose the Table or Query Where an Object's Data Comes From*, and click and highlight the *FRIENDS* table.
2. Now click and highlight the *Label Wizard* option, and then click *OK*.
3. On the first wizard screen, accept the default label type and dimensions. Click the Next> button.
4. On the second screen, select the font, weight and colour you want for the label text. Click the Next> button.
5. Choose fields for your label:

First Line

Select *Title*, and then click on the  button to lodge onto the label. Press Space on your keyboard.


Repeat this action with the *FirstName* field.

Select *LastName*, and then click . Now press Return on your keyboard.


Second Line

Select *Street*, and then click . Now press Return.

Third Line

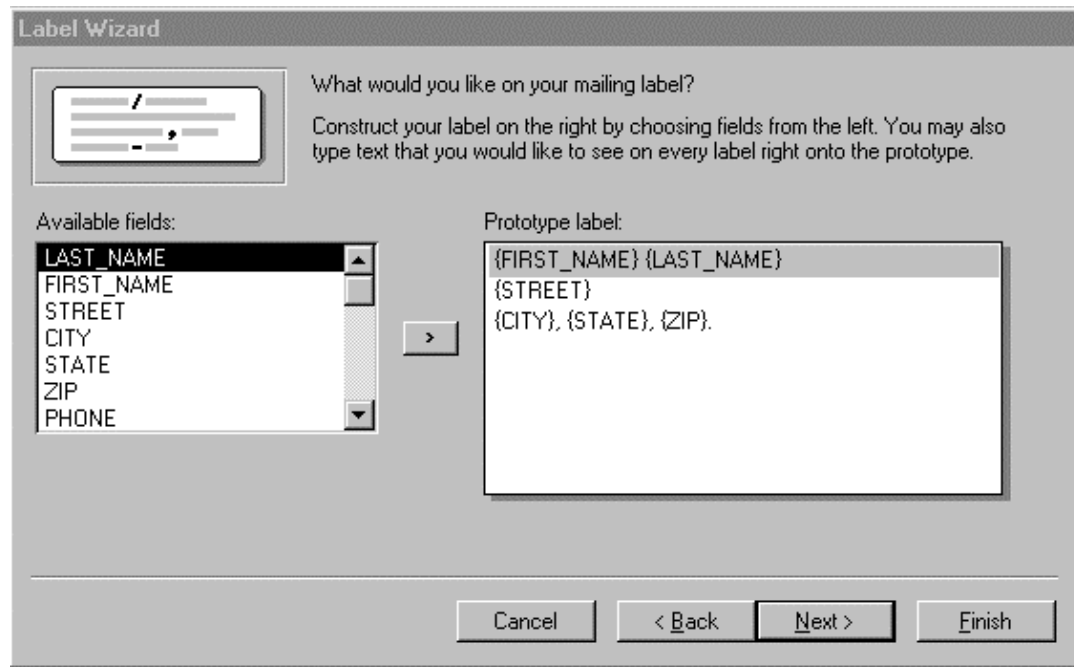
Select *City*, and then click . Add a comma (,) and then press Space.

Repeat this action with the *State* field.

Select *Zip*, and then click .

Your screen should now look like the one in Figure 5-69. Click the Next> button.

Figure 5-69



6. On the next report wizard screen, select the LastName field to sort by. Click the Next> button.
7. On the last wizard window, choose a name for your report (eg: *Tute6 Repo1*). Click the Finish button, and Access will generate your labels and display them in Print Preview mode.
8. Send your labels to print.
9. Press F11 to return to the Database Window, and then exit Access.

Tutorial For Database Case 6 Using Access 2.0



In this case, you will learn two important new skills: joining table files (creating relationships), and creating labels in reports.

Joining Tables

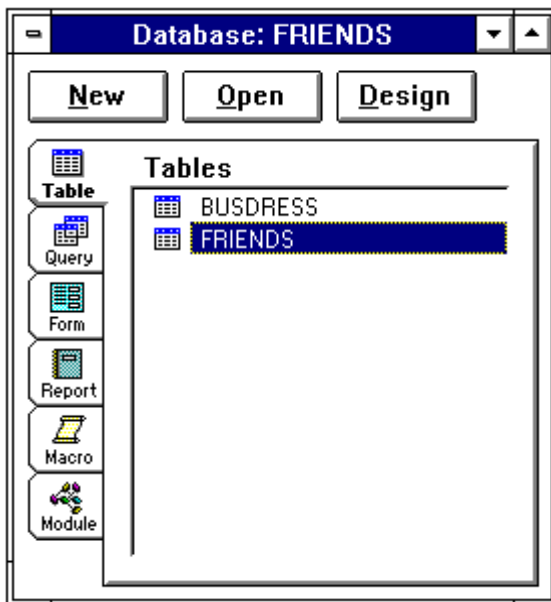
In many database applications, the joining of tables is a procedure that would require you to generate programming code. Access allows you to do this simply via the menu or toolbar, and to run the results of the join in a query (or a report). Access will also generate the SQL programming code for the procedure, so that you can see the effect of your actions.


Before you can join two tables, the related field must be present in both tables. While the field names do not have to be the same, for the join to work, the two fields must contain matching data within records. For this procedure, we will be using the FRIENDS database, and a second practice file BUSDRESS.DBF. The latter contains the last names and business addresses of persons in the FRIENDS database.

1. Load the practice database FRIENDS.MDB. Click on the *Table* object and then import the second practice database file BUSDRESS.DBF using FILE/IMPORT from the menu or by clicking on the *Import* toolbar button. Recap the tutorial in Chapter 4, if you are unsure about importing external files. BUSDRESS.DBF is a dBase file, so choose dBase III as the format during the import procedure. FRIENDS.MDB should now have two tables, and look similar to Figure 5-70.



Figure 5-70



2. From the Database Window, highlight and open the FRIENDS table, and then repeat this procedure for the BUSDRESS table. Compare the two tables, noting that both LastName fields contain identical data. Remember that the field names themselves do not have to be the same. Close the tables in turn by double clicking on the  box in the top left hand corner of each.

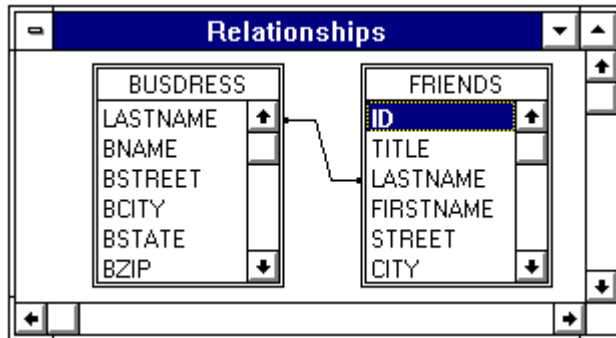
3. From the Database Window, create a relationship between the two tables. This will speed up the execution of the query we will create in a moment. Table relationships were last discussed in the tutorial for Chapter 4.

4. To create a relationship: click the *Relationships* button on the toolbar. From the Add Table, highlight the BUSDRESS table and click Add. Repeat this action for the FRIENDS table.



5. Highlight the LastName field in the BUSDRESS table and while holding down your left mouse button, drag this over to the LastName field in the FRIENDS table. Release the mouse button. Access generates a dynamic link (line) between the two known as an *equi-join*. An equi-join selects all the records from both tables that have the same value in the field selected as being common to both. Your Relationships window should look similar to Figure 5-71.

Figure 5-71



6. Click OK, and save the relationship by choosing FILE/SAVE LAYOUT or clicking on the *Save* toolbar button.


7. Press F11 to return to the Database Window.

You are now ready to create a query to generate the effect of the joining action.

8. From the Database


Window, click on the *Query* object, and click New. Now click on the New Query button.

9. From the *Add Table*, highlight the BUSDRESS table and click Add. Repeat this action for the FRIENDS table. Click Close. Notice that Access has remembered the table relationship, and has immediately established a join between the two LastName fields.

10. Select VIEW/SQL from the menu or click on the SQL toolbar button  to see the equivalent of this action in the Access Basic programming code:

```
SELECT DISTINCTROW
FROM BUSDRESS INNER JOIN FRIENDS ON BUSDRESS.LASTNAME =
FRIENDS.LASTNAME;
```



Click on the design button  to return to the query design window.

11. From the query window, select and drag the following fields down into the Field bar of the QBE Grid.

From FRIENDS

Title
FirstName
LastName

From BUSDRESS

BStreet
BCity
BState
BZip

Lastly, click on the *Sort*: bar in the LastName field, and choose Ascending order. Your query window should now look similar to Figure 5-72.

12. Click on the SQL button again, to see the effect this action has had on the Access Basic programming code:

```
SELECT DISTINCTROW FRIENDS.TITLE, FRIENDS.LASTNAME, FRIENDS.FIRSTNAME,
BUSDRESS.BSTREET, BUSDRESS.BCITY, BUSDRESS.BSTATE, BUSDRESS.BZIP
```

FROM BUSDRESS INNER JOIN FRIENDS ON BUSDRESS.LASTNAME =
FRIENDS.LASTNAME ORDER BY FRIENDS.LASTNAME;

Figure 5-72



13. Click on the **Run** query toolbar button, and view the resulting dynaset. Access has matched and merged the business address data in the BUSDRESS table with the name data in the FRIENDS table to create a composite display containing non-duplicative data from both (see Figure 5-73).

Figure 5-73

FRIENDS **Joined on** **BUSDRESS**


	TITLE	LASTNAME	FIRSTNAME	BSTREET	BCITY	BSTATE
▶	Dr	Drucker	Peter H.	345 Ohio St.	Cleveland	Ohio
	Mr	Whitney	Craig	345 Westin	Portland	Oregon
	Mr	Sitkin	Howard W.	4523 8th Avenue	Hartford	Conn
	Dr	Skalek	William F.	57 Morris	Seattle	Washington
	Prof	Salione	Phillip	459 Palm Drive	Springvale	California
	Mr	Fabian	James T.	98 Kansas Street	Burlington	Vermont
	Mr	Kohlman	Frank	856 Irvine	Irvine	California
	Mr	Tedesco	George R.	98 Lakes Drive	St. Paul	Minnesota
	Ms	Zito	Helen K.	23 Ocean Drive	Cambridge	Mass.
	Dr	Peterson	Jack S.	900 Lake Drive	Detroit	Michigan
	Dr	Nelson	Robert M.	75 Yorktown Circle	Yorktown	New York

Record: 1 of 11



14. Print the query if you wish to, or use as the basis for a new report.
15. Save the query (eg: *Tute6 Query1*), and press F11 to return to the Database Window.

How to Create Labels in Access


A report designed to print names and addresses on labels is a common feature of most database applications. The report feature in Access includes a powerful Mailing Label wizard, with sizes to match most common labels. Let's use the sample FRIENDS database to create a simple three line address label.

1. Load FRIENDS.MDB. Click on the Report object, and click New. From the New Reports box, click on the  button to the right of the *Select a Table/Query* Bar, and click and highlight the *FRIENDS* table.
2. Click on the *Report Wizards* button. Click on *Mailing Label*, and then click *OK*.
3. Choose fields for your label. Use the buttons below the field list to add spaces and punctuation to your label and to start each new line:



First Line

Select *Title*, and then click on the  button to lodge onto the report list. Click Space. Repeat this action with the *FirstName* field. Select *LastName*, and then click . Click Newline.

Second Line

Select *Street*, and then click . Click Newline.

Third Line

Select *City*, and then click . Click comma (,) and then click Space. Repeat this action with the *State* field. Select *Zip*, and then click .

Your screen should now look like the one in Figure 5-74. Click the Next> button.

Figure 5-74

Mailing Label Wizard

This Wizard creates standard Avery mailing labels. What do you want on your mailing label?

Select a field and then click the ">" button, type in appropriate text, or click a punctuation button.

Available fields:

- STREET
- CITY
- STATE
- ZIP

Label appearance:

TITLE·FIRSTNAME·LASTNAME
STREET
CITY·STATE·ZIP.

Buttons: > < Text > : , - . / Newline Space

Buttons: Hint Cancel < Back Next > Finish

4. On the next report wizard screen, select the LastName field to sort by. Click the Next> button.
5. On the third wizard screen, accept the default label type and dimensions. Click the Next> button.
6. On the final screen, select the font, weight and colour you want for the label text. Click the Finish button, and Access will generate your labels and display them in Print Preview mode.
7. Send your labels to print.
8. Save your new report (eg: *Tut61 Rep01*). Press F11 to return to the Database Window, and exit Access.