Database Case 10

Audio Sonics II

Problem: Create a complete system

Management skill: Control
Organize

Access Skills: Forms
Macros
Advanced Queries (Union)

Data Tables: SONIC_A
SONIC_B
SONIC_C

This case is a continuation of Solve it! Database Case 6. Refer to this case for background information relating to Case 10.

Brad Knox was feeling very pleased with himself as he gazed out of his office window at the busy Nashville streetscape. He had just finished his perusal of Anabasis' financial performance figures for the 2000-2001 year. His mail order music company had performed stunningly last year, with sales improving by nearly 18%. Knox had also noticed that the costs associated with postage and packaging of his mail order catalogs, and the maintenance of his mailing list had dropped significantly.

Knox believed that these results were largely due to the ability of the PC-based database mailing system his company had developed to provide carefully targeted mail outs for his various customer segments. Previously Knox had relied on blanket mailouts to all customers on his mailing list whenever he wanted to advertise new product offerings. Thanks to his new mailing system, his company was now able to tailor its mailouts so that catalogs were sent out only to customers with interests in the music areas, which were relevant to his new products.

Knox now wanted to complete the development of his new system by adding some enhancements and new features. He began to jot down his main requirements.

(a) A system that is easy to use.

The new system worked well, but it required a thorough understanding of the database package it had been created in to operate effectively. Knox was the only one who knew how the system worked and what it did, and he currently maintained the system himself. He did not regard

this as a productive way to spend his time, and wanted to pass over the operation and day-to-day maintenance of the system to his clerical staff.

Knox had to remember which query did what, and which report gave him the result formats he wanted. The system was efficient but it was not the kind of product you would want in the hands of untrained people. What was needed was a more robust and user-friendly version of the system.

Knox's objective is to produce a self-contained mailing system that could be operated by someone with little or no database knowledge. The system must be menu driven and modular in design. The menu should also execute automatically when the user opens the database, which contains the system (i.e.: the .MDB file). When a task option is selected from the menu, the menu will open the required object, and then execute that option. Each option should contain a facility to exit back to the menu.

(b) Integrate the Anabasis mailing list with that of Sonics Design.

Knox suspects that at least 25% of the customers on his mailing list are also customers who regularly purchase from Sonics Design, his brother's side of the business (refer back to *Case 6*). While the marketing side of Sonics Design has been handled by a PC-based database package for some time, the system is not menu driven and requires someone with a good understanding of databases to operate it. Knox and his brother would like to combine their mailing lists under one menu based system. They have agreed that while information unique to each of the businesses will be maintained separately, the names and addresses will be shared. Although the two divisions do business with many of the same organizations, Anabasis' customer contact names may often be different to the ones used by Sonics Design.

Brad Knox would like to test how well this procedure will work. His brother Trevor has provided him with a sample listing of Sonics Design customer names. This is provided on the *SolveIt!* data table SONIC_C. Brad wants to combine this file with his own listing (SONIC_A). The result of this procedure should show no duplicate records. The combined listings will then be incorporated into a menu option on the new system.

Knox needs your help. He does not have the time or expertise needed to create the menu driven mailing system required, and would like you to develop a prototype for him. Part of the job was completed in *SolveIt!* database Case 6, and you will need to use the database used for this case to complete Case 10.

Since this case is an extension of *SolveIt*! Case 6, it is recommended that you make a copy of the Case 6 files and save as Case 10, or to simply rename the Case 6 files as Case 10 before starting this case. As a first step, copy and save your Case 6 database with a new name.

Tasks

There are 6 tasks in this case.

1. Create a Main Menu form and appropriate command buttons, which allow the user the following executable options. Call this form MAINMENU.

1. Add a new customer.

2. Add a new company.

3. Produce a complete mailing list.

4. Exit the system.

Hint: Use the command button toolbox tool on the menu form for the first three options: These options will not initially be active. Use the form control wizards and the command button toolbox tool to set up an active Exit procedure for option four.

2. Create data entry forms for Options 1 and 2. Include Exit to Menu and Add New Record command buttons and test them thoroughly. Call these forms OPTION1 and OPTION2.

3. Devise a procedure that will execute the Main Menu automatically whenever the database is opened.

4. Create a procedure that will combine the customer name tables of Anabasis (SONIC_A) and Sonics Design (SONIC_C). There are a number of names, which appear in both tables. Your aim is to produce a listing with no duplicate records. The combined file should then be joined to the SONIC_B table to give a full listing of customers and their organizational details.

5. Create a report to display results for Task 4. The report should be sorted firstly by organization, and then by last name. The Country field should also be included. Incorporate this report as Option 3 on the Main Menu form

Hint: Use the existing report created for Task2/3 of Case 6, and simply reassign the report record source to reflect the changes.

*6. Extend the system even further to include the sales and number of sound recordings sold to each customer. Add the necessary fields and some sample data to the appropriate table. Then add a new option to the menu, which will calculate and display the total sales for Anabasis and the average price per recording.

Time Estimates (excluding task marked with *):

Expert: 2 hours
Intermediate: 3 hours
Novice: 5 hours

Tutorial For Database Case 10 Using Access 2000

Advanced Queries - the SQL-Specific Query

SQL (Structured Query Language) is a simple programming language used for querying, updating and managing relational databases. When you create a select or action query, Access automatically generates the equivalent SQL statement in background. You can view or edit this SQL statement by choosing SQL from the View menu in the Query window or clicking the SQL View toolbar button.

Some Access query types however, can only be created by writing an SQL statement. These queries are known as SQL-Specific queries. This group includes:

Union	*queries which combine fields from two or more tables or queries*
Pass-through	*queries which send commands directly to a database server*
Data-definition	*queries used to create or alter database tables in the current database*
Subqueries	*an SQL SELECT statement inside another select or action query*

SQL statements refer to expressions that define SQL commands such as SELECT, UPDATE or DELETE, and include clauses such as WHERE, GROUP BY and ORDER BY. SQL statements are typically used in the construct of queries and aggregate funtions.

The Union Query

You will need to use the Union SQL-Specific query type as part of the requirement for completing Task 1 of Case 10.

Using simple *SELECT ... FROM* SQL statements, *Union Queries* enable you to combine fields from two or more tables into one listing. In contrast, *Select queries, which* are based on a join, creates a dynaset only from those records whose related fields meet a specified condition. Commands and clauses commonly used in the creation of Union queries include:

SELECT *fieldlist*
FROM *tablenames* **IN** *database name*
WHERE *search conditions*
GROUP BY *fieldlist*
HAVING *search conditions*
ORDER BY *fieldlist*

To create a Union Query:

1. From the Database Window, click on the Queries object, and then click on the New button. This will display the New Query dialog box. Highlight the Design View option and then click OK.

2. Choose the Close button from the Show Table dialog box. Union queries do not use the Query Window/QBE Grid for their construction.

3. Choose QUERY/SQL SPECIFIC/UNION from the menu. Access displays the Union Query window.

4. Enter the appropriate SQL *SELECT....xx* statements needed for your Union Query. For example:

SELECT [Supplier Name], [City] FROM Suppliers UNION SELECT [Customer Name], [City] FROM Customers;	*Retrieves the names and cities of suppliers and customers from the Suppliers and Customers tables*
SELECT [Supplier Name], [City] FROM Suppliers WHERE [Country] = "Brazil" UNION SELECT [Customer Name], [City] FROM Customers WHERE [Country] = "Brazil" ORDER BY [City];	*Retrieves the names and cities of suppliers and customers located in Brazil, sorted by the City field*

5. Be sure to include a semi colon at the end of each statement (refer examples above) to complete the required Access syntax. Note also the use of square brackets to enclose field names. This is particularly important if you are using field names, which are composed of two or more words separated, by spaces.

6. Click on the Run Query toolbar button to execute and test your Union query. Save the query, and press the F11 key to exit back to the Database Window

Note: Unless you specify otherwise, a Union Query automatically removes duplicate records from the resulting listing. If you want to show all records, including duplicates, add the word ALL after the word UNION in your statement. For example:

SELECT [Supplier Name], [City] FROM Suppliers **UNION ALL** SELECT [Customer Name], [City] FROM Customers;	*Retrieves the names and cities of suppliers and customers from the Suppliers and Customers tables, and shows all records including duplicates*

Task 1 of Case 10 requires you to create a merged file of the customer records of Anabasis and Sonics Design. Contrast the difference between omitting and including ALL in the Union Query you need to create to complete this task.

> Caution: Don't convert an SQL-specific query to another type of query, such as a select query. If you do, you'll lose the SQL statement that you entered. *You can however use an SQL-specific query as part of a select query.*

Creating an AutoExec Macro

You can create a special macro that runs automatically whenever you open a Microsoft Access database. For example, you may wish to open certain tables and forms every time you open a database.

To create a macro that runs whenever you open a database

1. Create a macro.
2. Add the actions that you want this macro to perform.
3. Save the macro. Its name must be AutoExec.

AutoExec macros can be used to create a custom workspace, import data from other databases, or to execute tasks that you want to perform every time the database is opened. To prevent the AutoExec macro from running, hold down the Shift key while opening the database.

Tutorial For Database Case 10 Using Access 97

Advanced Queries - the SQL-Specific Query

SQL (Structured Query Language) is a simple programming language used for querying, updating and managing relational databases. When you create a select or action query, Access automatically generates the equivalent SQL statement in background. You can view or edit this SQL statement by choosing SQL from the View menu in the Query window or clicking the SQL View toolbar button.

Some Access query types however, can only be created by writing an SQL statement. These queries are known as SQL-Specific queries. This group includes:

Union	*queries which combine fields from two or more tables or queries*
Pass-through	*queries which send commands directly to a database server*
Data-definition	*queries used to create or alter database tables in the current database*
Subqueries	*an SQL SELECT statement inside another select or action query*

SQL statements refer to expressions that define SQL commands such as SELECT, UPDATE or DELETE, and include clauses such as WHERE, GROUP BY and ORDER BY. SQL statements are typically used in the construct of queries and aggregate funtions.

The Union Query

You will need to use the Union SQL-Specific query type as part of the requirement for completing Task 1 of Case 10.

Using simple *SELECT ... FROM* SQL statements, *Union Queries* enable you to combine fields from two or more tables into one listing. In contrast, *Select queries* which are based on a join, creates a dynaset only from those records whose related fields meet a specified condition. Commands and clauses commonly used in the creation of Union queries include:

SELECT *fieldlist*
FROM *tablenames* **IN** *database name*
WHERE *search conditions*
GROUP BY *fieldlist*
HAVING *search conditions*
ORDER BY *fieldlist*

To create a Union Query:

1. From the Database Window, click on the Queries object, and then click on the New button. This will display the New Query dialog box. Highlight the Design View option and then click OK.

2. Choose the Close button from the Show Table dialog box. Union queries do not use the Query Window/QBE Grid for their construction.

3. Choose QUERY/SQL SPECIFIC/UNION from the menu. Access displays the Union Query window.

4. Enter the appropriate SQL *SELECT....xx* statements needed for your Union Query. For example:

SELECT [Supplier Name], [City] FROM Suppliers UNION SELECT [Customer Name], [City] FROM Customers;	*Retrieves the names and cities of suppliers and customers from the Suppliers and Customers tables*
SELECT [Supplier Name], [City] FROM Suppliers WHERE [Country] = "Brazil" UNION SELECT [Customer Name], [City] FROM Customers WHERE [Country] = "Brazil" ORDER BY [City];	*Retrieves the names and cities of suppliers and customers located in Brazil, sorted by the City field*

5. Be sure to include a semi colon at the end of each statement (refer examples above) to complete the required Access syntax. Note also the use of square brackets to enclose field names. This is particularly important if you are using field names, which are composed of two or more words separated, by spaces.

6. Click on the Run Query toolbar button to execute and test your Union query. Save the query, and press the F11 key to exit back to the Database Window

Note: Unless you specify otherwise, a Union Query automatically removes duplicate records from the resulting listing. If you want to show all records, including duplicates, add the word ALL after the word UNION in your statement. For example:

SELECT [Supplier Name], [City] FROM Suppliers **UNION ALL** SELECT [Customer Name], [City] FROM Customers;	*Retrieves the names and cities of suppliers and customers from the Suppliers and Customers tables, and shows all records including duplicates*

Task 1 of Case 10 requires you to create a merged file of the customer records of Anabasis and Sonics Design. Contrast the difference between omitting and including ALL in the Union Query you need to create to complete this task.

> Caution: Don't convert an SQL-specific query to another type of query, such as a select query. If you do, you'll lose the SQL statement that you entered. *You can however use an SQL-specific query as part of a select query.*

Creating an AutoExec Macro

You can create a special macro that runs automatically whenever you open a Microsoft Access database. For example, you may wish to open certain tables and forms every time you open a database.

To create a macro that runs whenever you open a database

1. Create a macro.
2. Add the actions that you want this macro to perform.
3. Save the macro. Its name must be AutoExec.

AutoExec macros can be used to create a custom workspace, import data from other databases, or to execute tasks that you want to perform every time the database is opened. To prevent the AutoExec macro from running, hold down the Shift key while opening the database.