SeQueL 3 – Queries - JOIN from Afar!

by Clark Anderson

Please re-JOIN me in the continuation of the adventures in Structured Query Language. Last time in **"SeQueL 2 – Queries - JOIN in Now!"**, we explored a lot of ways to select data records joined together from more than one table. Now it is time to broaden our horizons beyond one database.

Microsoft Access provides a convenient way to start. In the Database Window, be sure that the Tables tab is selected. On the Access Menu Bar, select File/Get External Data/Link Tables… . Locate, Select and Link the other database (e.g.:D:\...\LookUp.mdb). A list of the available tables will be presented. I chose LUStates. In this list you may see numerous other tables that have names beginning with Msys. They are part of Microsoft Access. We shall leave them alone, for now. After this process, you will notice the "linked" table listed in with your other tables. It has an arrowhead to the left of its icon to indicate that it is linked rather than "local". In the earlier version, Access 2.0, Linked tables were referred to as "Attached".

While a table is linked, you can use it like it was part of the original database. The Query design tools in Access are pretty useful. You can get started by selecting the Queries tab on the Database Window. Clicking New and selecting Design View will open the Query Design Window for a Select query and show a list of tables. I selected LUStates, clicked Add, then Close. The Query Design Window shows the table with its field names. The primary key field names are shown in bold letters. The asterisk at the top of the table's list of fields represents all of the fields in the table.

The bottom part of the Query Design Window contains a grid for specifying the different parts of your query. You can doubleclick on a field name or drag a field name to a column in the grid. The Field row of the grid specifies the Field name. The Table row specifies the name of the table the field is FROM. The Sort row is used to define the ORDER BY clause of the query. The checkboxes in the Show row specify the fields to appear in the SELECTed list of fields that will be retrieved by the query. The Criteria row is used to build the WHERE clause. To build this query, I typed IN ('CO','CT','WA','KS','ON','FL','AB') into the Criteria row of the StateCode column. When I am defining constants in a query I prefer to surround them with apostrophes instead of quotes. Now you can right click in the Query Design Window title bar and select SQL View from the pop-up menu. This resulting SQL query was displayed.

```
SELECT LUStates.StateCode, LUStates.StateName, LUStates.Country
FROM LUStates
WHERE (((LUStates.StateCode) In ('CO','CT','WA','KS','ON','FL','AB')));
```

Again a right click on the Query Design Window title bar let me select Datasheet View:

StateCode	StateName	Country
CO	Colorado	USA
CT	Connecticut	USA
FL	Florida	USA
KS	Kansas	USA
WA	Washington	USA
AB	Alberta	Canada
ON	Ontario	Canada

I returned to the SQL View and removed the table name and separating dot, (e.g.: "LUStates."), from the field names in the SELECT clause and the WHERE clause. There are no other tables in this query. I also removed the extra parentheses from the WHERE clause. The only ones that were really needed were those that are a part of the IN phrase.

```
SELECT StateCode, StateName, Country FROM LUStates
WHERE StateCode IN ('CO','CT','WA','KS','ON','FL','AB');
```

Next, I inserted into the FROM clause the specific location of the External database containing the table, LUStates. The format for this includes a left bracket, the full path, the database file name, a right bracket and a dot followed by the table name. For this query the specific table definition accomplishes the same results as having the table linked.

```
SELECT StateCode, StateName, Country
FROM [D:\SCRATCH\EMAIL\ACCESSUG\Articles\SQLJoin3\LookUp.mdb].LUStates
WHERE StateCode IN ('CO','CT','WA','KS','ON','FL','AB');
```

For another example, I created another new query. This time I selected two tables: CoffeeHouses and LUStates

CoffeeHouses Table

Shop	City	State
Moe's	Bloomfield	CT
Real Coffee	Seattle	WA
Cafe Luna	Longmont	CO
Expresso Roma	Boulder	CO
Central Café	Lyons	KS
Aggie's Diner	Ottawa	ON

I dragged the CoffeeHouses.State field name and dropped it on the LUStates.StateCode field name. This created an INNER JOIN from the CoffeeHouses table to the LUStates table. The results if this query will only include rows where CoffeeHouses.State = LUStates.StateCode. I selected Shop and City from the CoffeeHouses table and I selected StateName from the LUStates table. Then I went back to SQL View.

```
SELECT CoffeeHouses.Shop, CoffeeHouses.City, LUStates.StateName
FROM CoffeeHouses INNER JOIN LUStates ON CoffeeHouses.State = LUStates.StateCode;
```

JOIN clauses are an extended part of the FROM clause, because they help specify all of the tables that are needed in the query. I inserted into the JOIN clause the specific location of the External database containing the table, LUStates.

```
SELECT CoffeeHouses.Shop, CoffeeHouses.City, LUStates.StateName
FROM CoffeeHouses INNER JOIN [D:\SCRATCH\EMAIL\ACCESSUG\Articles\SQLJoin3\LookUp.mdb].LUStates ON
CoffeeHouses.State = LUStates.StateCode;
```

Both queries have the same result.

Shop	City	StateName
Moe's	Bloomfield	Connecticut
Real Coffee	Seattle	Washington
Cafe Luna	Longmont	Colorado
Expresso Roma	Boulder	Colorado
Central Café	Lyons	Kansas
Aggie's Diner	Ottawa	Ontario

Again the query can be simplified by removing the table names from the SELECT clause, as long as there are no ambiguities. (e.g.: selected field names existing in multiple tables) This also applies to WHERE clauses, but JOIN clauses cannot tolerate this kind of cleanup. Often a lot of parentheses can be cleaned out of WHERE clauses. They do need enough to clearly define their logical intent.

```
SELECT Shop, City, StateName
FROM CoffeeHouses INNER JOIN [D:\SCRATCH\EMAIL\ACCESSUG\Articles\SQLJoin3\LookUp.mdb].LUStates ON
CoffeeHouses.State = LUStates.StateCode;
```

Let us now build an Update Query. In creating this new query in the Design View, I selected the same two tables: CoffeeHouses and LUStates. A right click on the Query design window title bar popped up the menu for me to select Query Type/Update Query. To create the same INNER JOIN, I dragged the CoffeeHouses.State field name and dropped it on the LUStates.StateCode field name. In the Update Query Design View, the Select Query's Row Labels (Sort: and Show:) have been replaced by (Update To:). When you click on a cell in the Field row, the down arrow button appears on the right. When you click that button the drop-down list of fields appears. I chose CoffeeHouses.State. This filled in the Field and Table cells in this column. Microsoft Access expects you to type a constant value in the Update To cell, but I entered [LUStates].[StateName]. The brackets around the JOINed table name and field name are needed to indicate that it is NOT a constant. The update information will be drawn from that field in the JOINed table. We can add a WHERE clause by entering LEN(State) in the Field cell of another column and 2 in the Criteria cell of that column.

```
UPDATE CoffeeHouses INNER JOIN LUStates ON CoffeeHouses.State = LUStates.StateCode
SET CoffeeHouses.State = [LUStates].[StateName]
WHERE (((Len([State]))=2));
```

The WHERE clause is not needed here, because the INNER JOIN to LUStates.StateCode efectively restricts the length to 2 characters. I wanted you to see how a WHERE clause could be done in an Update Query.

```
UPDATE CoffeeHouses INNER JOIN LUStates ON CoffeeHouses.State = LUStates.StateCode
SET CoffeeHouses.State = [LUStates].[StateName];
```

You might think it easier to edit the raw SQL:

Replace	SELECT field1, field2, … FROM
With	UPDATE
Add	SET field1 = value1, field2 = value2, …

The same kind of "cleanup" and specific location of the External database works here:

```
UPDATE CoffeeHouses INNER JOIN [D:\SCRATCH\EMAIL\ACCESSUG\Articles\SQLJoin3\LookUp.mdb].LUStates ON
CoffeeHouses.State = LUStates.StateCode
SET State = [LUStates].[StateName];
```

By changing the INNER JOIN to the lookup table from StateCode to StateName and also reversing the SET clause value, I made a query to set the CoffeeHouses.State field back to its original State Codes.

```
UPDATE CoffeeHouses INNER JOIN [D:\SCRATCH\EMAIL\ACCESSUG\Articles\SQLJoin3\LookUp.mdb].LUStates ON
CoffeeHouses.State = LUStates.StateName
SET State = [LUStates].[StateCode];
```

My use of specific location for an External database goes beyond academic interest. Most of my SQL queries are inserted into Visual Basic programs using Access databases that can be located many different places. The programs do know where the databases are located, so they fill in the location information before using the query.

I believe this will stir up more of your own ideas until our next adventure in SQL.