SeQueL 6 – Queries – Need a UNION?

by Clark Anderson

UNION queries are off the beaten path. I can remember one very good use for a pretty complicated UNION query. I used it to create a Snapshot Recordset for populating a DropDown List. MS Access 97 Definition: "UNION queries combine corresponding fields from two or more tables or queries into one field". The result cannot be edited.

The bad news is that there is usually no help from a wizard or direct support in the Design View. With a UNION query your choices are SQL View or Datasheet View.

The good news is the UNION query is made of 2 or more SELECT statements. Individually, they work quite well in the Design View or the wizard.

Here is s simple SELECT query for starters. It provides a list of bagels ordered.

```
SELECT BagelsOrdered.BagelID,
BagelsOrdered.Name, BagelsOrdered.Quantity
FROM BagelsOrdered
```

BagelID	Name	Quantity
1	Plain	240
2	Egg	120
5	Whole Wheat	120
3	Everything	60

This query provides identical columns from a list of all bagels that might be ordered. Notice that the Quantity has been faked with a constant value of zero

```
SELECT LUBagel.BagelID, LUBagel.Name, 0 AS
Quantity
FROM LUBagel
```

BagelID	Name	Quantity
1	Plain	0
2	Egg	0
3	Everything	0
4	Sesame Seed	0
5	Whole Wheat	0
6	Cinnamon Raisin	0
7	Poppy Seed	0

Now, a drum roll please!

A simple copy and paste in SQL View brings these two simple queries together into a UNION query!

I simply insert the UNION command between the two SELECT statements and VIOLA!

```
SELECT LUBagel.BagelID, LUBagel.Name, 0 AS
Quantity
FROM LUBagel
UNION
SELECT BagelsOrdered.BagelID,
BagelsOrdered.Name, BagelsOrdered.Quantity
FROM BagelsOrdered
ORDER BY BagelID, Quantity
```

BagelID	Name	Quantity
1	Plain	0
1	Plain	240
2	Egg	0
2	Egg	120
3	Everything	0
3	Everything	60
4	Sesame Seed	0
5	Whole Wheat	0
5	Whole Wheat	120
6	Cinnamon Raisin	0
7	Poppy Seed	0

This bagel order list has a few redundant entries, so I have modified the second query with a LEFT JOIN statement to only list the records that do not appear in the BagelsOrdered query. The LEFT JOIN will include all records in the LUBagel table and any matching records from the BagelsOrdered table. The WHERE clause eliminates the records that found a match in the BagelsOrdered table.

```
SELECT LUBagel.BagelID, LUBagel.Name, 0 AS
Quantity
FROM LUBagel LEFT JOIN BagelsOrdered ON
LUBagel.BagelID = BagelsOrdered.BagelID
WHERE BagelsOrdered.BagelID Is Null
```

BagelID	Name	Quantity
4	Sesame Seed	0
6	Cinnamon Raisin	0
7	Poppy Seed	0

Copy and paste this SELECT – JOIN query with the BagelsOrdered SELECT query and we have a UNION query that has been improved to have a cleaned up Bagel Order list with no repeated bagels.

```
SELECT LUBagel.BagelID, LUBagel.Name, 0 AS
Quantity
FROM LUBagel LEFT JOIN BagelsOrdered ON
LUBagel.BagelID = BagelsOrdered.BagelID
WHERE BagelsOrdered.BagelID Is Null
UNION
SELECT BagelsOrdered.BagelID,
BagelsOrdered.Name, BagelsOrdered.Quantity
FROM BagelsOrdered
ORDER BY BagelID
```

BagelID	Name	Quantity
1	Plain	240
2	Egg	120
3	Everything	60
4	Sesame Seed	0
5	Whole Wheat	120
6	Cinnamon Raisin	0
7	Poppy Seed	0

Now, to demonstrate a point I will create a silly query. It provides a list of Bagel Names, Coffee House Shop Names and City Names. Nobody would want this hodge podge of a list but the UNION query only cares that the fields provided by each SELECT statement are the same data type and size. In this query I have assured with the LEFT function that all returned values are 20 character text strings.

```
SELECT  LUBagel.Name
FROM LUBagel
UNION
SELECT LEFT(CoffeeHouses.Shop,20) AS Name
FROM CoffeeHouses
UNION
SELECT LEFT(CoffeeHouses.City,20) AS Name
FROM CoffeeHouses
ORDER BY Name
```

Name
Aggie's Diner
Bloomfield
Boulder
Cafe Luna
Central Caf,
Cinnamon Raisin
Egg
Everything
Expresso Roma
Longmont
Lyons
Moe's
Ottawa
Plain
Poppy Seed
Real Coffee
Seattle
Sesame Seed
Whole Wheat

If you were combining integers with Long Integers, you must, in the SELECT statements, convert them all to Long Integers. If you were combining Date fields with Text fields containing dates you must do the conversions. Something like:

```
SELECT LngNumDate.LngNum AS ID,
LngNumDate.Date AS RealDate
FROM LngNumDate
UNION
SELECT CLNG(IntNumTxtDte.IntNum) AS ID,
CDATE(IntNumTxtDte.TxtDate) AS RealDate
FROM IntNumTxtDte
ORDER BY ID
```

I have memories of past trouble when I did not make all corresponding UNION fields equivalent data types. I tested the query by removing the functions. It 'seemed' to work. Then I changed the ORDER BY from ID to RealDate and got the 'Data Type mismatch in criteria expression' error message.

Please note that the ORDER BY command is placed at the end of the entire UNION query and any JOIN and WHERE clauses are with the appropriate individual SELECT statements.

All of these queries have been built and tested in Access 97.

Now you know what a UNION query is and you will know how to put one together when the need arises. If your schedule permits, experiment, push the boundaries!