

How Access Stores Queries – Part 1

This will be a significantly expanded version of the article first published on 3 Feb 2019 at <https://www.isladogs.co.uk/explaining-queries/>

1. Introduction

Access uses the **MSysQueries** system table in conjunction with other **system tables** to display the query structure in the **query design window**.

The query design is **optimised** by Access so it runs in the **most efficient way possible**

The stored data for each query is automatically replaced each time a query is used to ensure any design changes are saved.

IMPORTANT

System tables are used by Access to make databases function correctly

Some system tables can be viewed & a few can be edited

But that doesn't mean you should do soUNLESS YOU ARE ABSOLUTELY SURE WHAT YOU ARE DOING

Altering one table may have 'knock on' effects on other tables

Incorrectly editing system tables may corrupt your database or prevent you opening it

Anyway, having made that point, I'll continue...

Although it is ALWAYS important to take great care when viewing system tables, the **MSysQueries** table is **READ ONLY** so no damage can be done. Even so, it is better to create a **query** for this purpose than view the **system table** directly.

Attribute	Expression	Flag	LvExtra	Name1	Name2	ObjectId	Order
		0				-2147483630	
1		1				-2147483630	
3		0				-2147483630	
5				MSysObjects		-2147483630	
5				MSysQueries		-2147483630	
6	MSysObjects.Name	0				-2147483630	
6	MSysQueries.Attribute	0				-2147483630	
6	MSysQueries.Flag	0				-2147483630	
6	MSysQueries.Expression	0				-2147483630	
6	MSysQueries.Name1	0				-2147483630	
6	MSysQueries.Name2	0				-2147483630	
7	MSysObjects.Id = MSysQueries.ObjectId	1		MSysObjects	MSysQueries	-2147483630	
8	((MSysObjects.Flags <> 3))					-2147483630	
11	MSysObjects.Name					-2147483630	
11	MSysQueries.Attribute					-2147483630	
11	MSysQueries.Flag					-2147483630	
255						-2147483630	

The easiest way to interrogate this table is in conjunction with the **MSysObjects** system table (also READ ONLY)
Not all fields are required for this purpose:

Field:	Name	Attribute	Flag	Expression	Name1	Name2	Flags
Table:	MSysObjects	MSysQueries	MSysQueries	MSysQueries	MSysQueries	MSysQueries	MSysObjects
Sort:	Ascending	Ascending	Ascending				
Show:	☒	☒	☒	☒	☒	☒	☐
Criteria:							<> 3
or:							

```
SELECT MSysObjects.Name, MSysQueries.Attribute, MSysQueries.Flag, MSysQueries.Expression, MSysQueries.Name1,
MSysQueries.Name2
FROM MSysObjects INNER JOIN MSysQueries ON MSysObjects.Id = MSysQueries.ObjectId
WHERE (((MSysObjects.Flags)<>3))
ORDER BY MSysObjects.Name, MSysQueries.Attribute, MSysQueries.Flag;
```

NOTE: Flags value 3 is used for TEMP queries used with form and report record sources so is OMITTED here

qryMSysQueries						
Name	Attribute	Flag	Expression	Name1	Name2	
qryMSysQueries	0	0				
qryMSysQueries	1	1				
qryMSysQueries	3	0				
qryMSysQueries	5			MSysObjects		
qryMSysQueries	5			MSysQueries		
qryMSysQueries	6	0	MSysQueries.Name1			
qryMSysQueries	6	0	MSysObjects.Name			
qryMSysQueries	6	0	MSysQueries.Attribute			
qryMSysQueries	6	0	MSysQueries.Expression			
qryMSysQueries	6	0	MSysQueries.Name2			
qryMSysQueries	6	0	MSysQueries.Flag			
qryMSysQueries	7	1	MSysObjects.Id = MSysQueries.ObjectId	MSysObjects	MSysQueries	
qryMSysQueries	8		((MSysObjects.Flags)<>3))			
qryMSysQueries	11		MSysObjects.Name			
qryMSysQueries	11		MSysQueries.Attribute			
qryMSysQueries	11		MSysQueries.Flag			
qryMSysQueries	255					

2. The MSysQueries table explained

The table below explains the meaning of each field for different query types.

The Expression field is blank unless stated otherwise:

Attribute	Purpose	Expression Info	Flag	Comments
0	Start record		0	Always blank
1	OPERATION (QUERY TYPE) Flag indicates query type		1	SELECT
			2	INSERT INTO (Make table) Table name in Name1; External db path in Name2
			3	INSERT (Append); Table name in Name1
			4	UPDATE
			5	DELETE
			6	TRANSFORM (Crosstab)
			7	Data Definition
			8	Pass through
			9	UNION
2	PARAMETER Flag indicates datatype	Parameter fields	0	Value
			1	dbBoolean
			2	dbByte
			3	dbInteger
			4	dbLong
			5	dbCurrency
			6	dbSingle
			7	dbDouble
			8	dbDate
			9	dbBinary
			10	dbText
			11	dbLongBinary (OLE Object / Bitmap)
			12	dbMemo (Long Text)
			15	dbGUID (Replication ID)
			16	dbBigInt (Decimal)
			0	No special attributes
			1	Output All Fields (for SELECT queries) TEMP queries (form/report record sources) UNION ALL (for UNION queries)
3	OPTION (UNION / DISTINCT / TOP) Flag indicates special attributes (Values can be cumulative)		2	DISTINCT
			3	UNION
			4	WITH OWNERACCESS OPTION
			8	DISTINCTROW
			9	TEMP queries (form/report record sources)
			16	TOP
			18	DISTINCT TOP (2+16)
			24	DISTINCTROW TOP (8+16)
			48	TOP PERCENT
			50	DISTINCT TOP PERCENT (2+48)
			56	DISTINCTROW TOP PERCENT (8+48)
4	Connection string	Connection string to external table		Query to external database
5	TABLES / UNION / SUBQUERY			Name1 = table ; Name2 = alias (if any)
		Individual SQL statements in UNION queries		Name2 – identifier for each UNION segment
		Subquery SQL		Name2 – subquery table alias
6	FIELDS / EXPRESSIONS	Table & field names for SELECT queries	0	Aliases in Name1 field
		Update to value in UPDATE queries	0	Field name in Name2
		Append field in APPEND queries	0	Append to field in Name2
		Aggregate expressions	0	
		FROM expression in DELETE queries	0	
		TRANSFORM in CROSSTAB (Value)	0	Alias in Name2
		PIVOT in CROSSTAB (Column Heading)	1	
		Row Headings in CROSSTAB	2	
		Append field in APPEND VALUES queries	-32768	Append to field in Name2
7	JOINS	Inner Join expression	1	Join table names in Name1 & Name2
		Left Join expression	2	Join table names in Name1 & Name2
		Right Join expression	3	Join table names in Name1 & Name2
8	WHERE	Filter expression		
9	GROUP BY	Aggregate query group table/field names	0	
		Crosstab aggregate query PIVOT (Column heading)	1	
		Crosstab aggregate query (Row headings)	2	
10	HAVING	Filter expression in Aggregate query		
11	ORDER BY	Sort order table/field names		D or d in Name1 if descending
12	Complex datatypes & temp queries		1	Query with long text field version history data
			2	Query with MVF / attachment data TEMP queries created using wizard
255	End record			Always blank

<https://docs.microsoft.com/en-us/office/client-developer/access/desktop-database-reference/datatypeenum-enumeration-dao>

SO article

<https://stackoverflow.com/questions/17329223/what-does-the-data-in-msysqueries-mean/54494758#54494758>

The attached database includes several queries to illustrate how the **MSysQueries** table works:

3. SELECT query – INNER JOIN & ORDER BY

Field:	CurrencyCode	Currency	Base	Date	Currency	Rate
Table:	tblCurrencies	tblCurrencies	tblCurrencyExchange	tblCurrencyExchange	tblCurrencyExchange	tblCurrencyExchange
Sort:	Ascending			Descending		
Show:	☒	☒	☒	☒	☒	☒
Criteria:						
or:						

```
SELECT tblCurrencies.CurrencyCode, tblCurrencies.Currency, tblCurrencyExchange.Base, tblCurrencyExchange.Date,
tblCurrencyExchange.Currency, tblCurrencyExchange.Rate
FROM tblCurrencies INNER JOIN tblCurrencyExchange ON tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency
ORDER BY tblCurrencies.CurrencyCode, tblCurrencyExchange.Date DESC;
```

Attribute 1 has Flag = 1 indicating a SELECT query

Attribute 5 shows table names, 6 has field names, 7 has join info, 11 has ORDER BY info

Name	Attribute	Flag	Expression	Name1	Name2
qryCurrencyExchangeINNER	0	0			
qryCurrencyExchangeINNER	1	1			
qryCurrencyExchangeINNER	3	0			
qryCurrencyExchangeINNER	5			tblCurrencyExc	
qryCurrencyExchangeINNER	5			tblCurrencies	
qryCurrencyExchangeINNER	6	0	tblCurrencyExchange.Rate		
qryCurrencyExchangeINNER	6	0	tblCurrencyExchange.Currency		
qryCurrencyExchangeINNER	6	0	tblCurrencyExchange.Date		
qryCurrencyExchangeINNER	6	0	tblCurrencyExchange.Base		
qryCurrencyExchangeINNER	6	0	tblCurrencies.Currency		
qryCurrencyExchangeINNER	6	0	tblCurrencies.CurrencyCode		
qryCurrencyExchangeINNER	7	1	tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency	tblCurrencies	tblCurrencyExchange
qryCurrencyExchangeINNER	11		tblCurrencyExchange.Date	D	
qryCurrencyExchangeINNER	11		tblCurrencies.CurrencyCode		
qryCurrencyExchangeINNER	255				

4. SELECT query – INNER Join (Filtered)

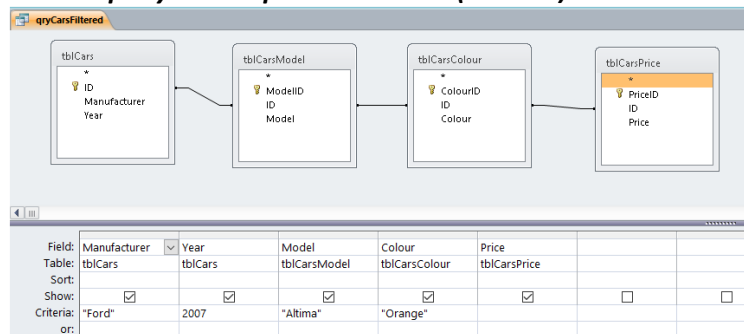
Field:	CurrencyCode	Currency	Base	Date	Currency	Rate
Table:	tblCurrencies	tblCurrencies	tblCurrencyExchange	tblCurrencyExchange	tblCurrencyExchange	tblCurrencyExchange
Sort:	Ascending					
Show:	☒	☒	☒	☒	☒	☒
Criteria:			"GBP"	#14/12/2018#		
or:						

```
SELECT tblCurrencies.CurrencyCode, tblCurrencies.Currency, tblCurrencyExchange.Base, tblCurrencyExchange.Date,
tblCurrencyExchange.Currency, tblCurrencyExchange.Rate
FROM tblCurrencies INNER JOIN tblCurrencyExchange ON tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency
WHERE (((tblCurrencyExchange.Base)="GBP") AND ((tblCurrencyExchange.Date)=#12/14/2018#))
ORDER BY tblCurrencies.CurrencyCode;
```

Attribute 8 has the WHERE clause info

Name	Attribute	Flag	Expression	Name1	Name2
qryCurrencyExchangeINNERFiltered	0	0			
qryCurrencyExchangeINNERFiltered	1	1			
qryCurrencyExchangeINNERFiltered	3	0			
qryCurrencyExchangeINNERFiltered	5			tblCurrencyExc	
qryCurrencyExchangeINNERFiltered	5			tblCurrencies	
qryCurrencyExchangeINNERFiltered	6	0	tblCurrencyExchange.Rate		
qryCurrencyExchangeINNERFiltered	6	0	tblCurrencyExchange.Currency		
qryCurrencyExchangeINNERFiltered	6	0	tblCurrencyExchange.Date		
qryCurrencyExchangeINNERFiltered	6	0	tblCurrencyExchange.Base		
qryCurrencyExchangeINNERFiltered	6	0	tblCurrencies.Currency		
qryCurrencyExchangeINNERFiltered	6	0	tblCurrencies.CurrencyCode		
qryCurrencyExchangeINNERFiltered	7	1	tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency	tblCurrencies	tblCurrencyExchange
qryCurrencyExchangeINNERFiltered	8		((tblCurrencyExchange.Base)="GBP") AND ((tblCurrencyExchange.Date)=#12/14/2018#)		
qryCurrencyExchangeINNERFiltered	11		tblCurrencies.CurrencyCode		
qryCurrencyExchangeINNERFiltered	255				

5. SELECT query – Multiple INNER Joins (Filtered)

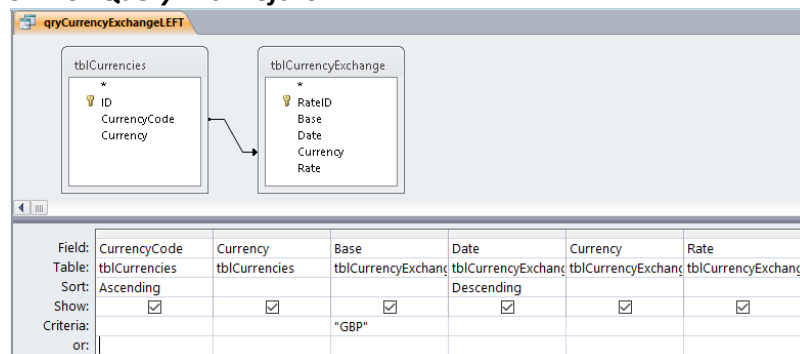


```
SELECT DISTINCT tblCars.Manufacturer, tblCars.Year, tblCarsModel.Model, tblCarsColour.Colour, tblCarsPrice.Price
FROM ((tblCars INNER JOIN tblCarsModel ON tblCars.ID = tblCarsModel.ID) INNER JOIN tblCarsColour ON
tblCarsModel.ID = tblCarsColour.ID) INNER JOIN tblCarsPrice ON tblCarsColour.ID = tblCarsPrice.ID
WHERE (((tblCars.Manufacturer)="Ford") AND ((tblCars.Year)=2007) AND ((tblCarsModel.Model)="Altima") AND
((tblCarsColour.Colour)="Orange"));
```

Attribute 3 has Flag = 2 indicating unique values (DISTINCT)

Name	Attribute	Flag	Expression	Name1	Name2
qryCarsFiltered	0	0			
qryCarsFiltered	1	1			
qryCarsFiltered	3	2			
qryCarsFiltered	5			tblCars	
qryCarsFiltered	5			tblCarsColour	
qryCarsFiltered	5			tblCarsModel	
qryCarsFiltered	5			tblCarsPrice	
qryCarsFiltered	6	0	tblCarsModel.Model		
qryCarsFiltered	6	0	tblCars.Year		
qryCarsFiltered	6	0	tblCarsColour.Colour		
qryCarsFiltered	6	0	tblCarsPrice.Price		
qryCarsFiltered	6	0	tblCars.Manufacturer		
qryCarsFiltered	7	1	tblCars.ID = tblCarsModel.ID	tblCars	tblCarsModel
qryCarsFiltered	7	1	tblCarsModel.ID = tblCarsColour.ID	tblCarsModel	tblCarsColour
qryCarsFiltered	7	1	tblCarsColour.ID = tblCarsPrice.ID	tblCarsColour	tblCarsPrice
qryCarsFiltered	8		((tblCars.Manufacturer)="Ford") AND ((tblCars.Year)=2007) AND ((tblCarsModel.Model)="Altima") AND ((tblCarsColour.Colour)="Orange"))		
qryCarsFiltered	255				

6. SELECT Query with Left Join



```
SELECT tblCurrencies.CurrencyCode, tblCurrencies.Currency, tblCurrencyExchange.Base, tblCurrencyExchange.Date,
tblCurrencyExchange.Currency, tblCurrencyExchange.Rate
FROM tblCurrencies LEFT JOIN tblCurrencyExchange ON tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency
WHERE (((tblCurrencyExchange.Base)="GBP"))
```

```
ORDER BY tblCurrencies.CurrencyCode, tblCurrencyExchange.Date DESC;
```

Attribute 7 has Flag=2 indicating a left join

qryMSysQueries					
Name	Attribute	Flag	Expression	Name1	Name2
qryCurrencyExchangeLEFT	0	0			
qryCurrencyExchangeLEFT	1	1			
qryCurrencyExchangeLEFT	3	0			
qryCurrencyExchangeLEFT	5			tblCurrencies	
qryCurrencyExchangeLEFT	5			tblCurrencyExc	
qryCurrencyExchangeLEFT	6	0	tblCurrencyExchange.Date		
qryCurrencyExchangeLEFT	6	0	tblCurrencies.CurrencyCode		
qryCurrencyExchangeLEFT	6	0	tblCurrencyExchange.Base		
qryCurrencyExchangeLEFT	6	0	tblCurrencyExchange.Currency		
qryCurrencyExchangeLEFT	6	0	tblCurrencyExchange.Rate		
qryCurrencyExchangeLEFT	6	0	tblCurrencies.Currency		
qryCurrencyExchangeLEFT	7	2	tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency	tblCurrencies	tblCurrencyExchange
qryCurrencyExchangeLEFT	8		((tblCurrencyExchange.Base)="GBP"))		
qryCurrencyExchangeLEFT	11		tblCurrencies.CurrencyCode		
qryCurrencyExchangeLEFT	11		tblCurrencyExchange.Date	D	
qryCurrencyExchangeLEFT	255				

7. *SELECT TOP* Query

[illegible]

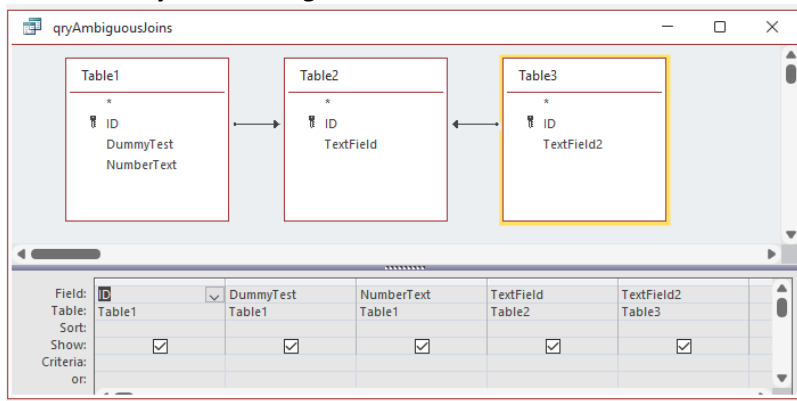
```
SELECT TOP 20 Count(WhoseOn.LogInID) AS TotalLogins, WhoseOn.TeacherID
FROM WhoseOn
WHERE (((WhoseOn.TimeOn)>=GetAcYearStart()))
GROUP BY WhoseOn.TeacherID
HAVING ((WhoseOn.TeacherID)<>"****")
ORDER BY Count(WhoseOn.LogInID) DESC, WhoseOn.TeacherID;
```

Attribute 3 has Flag = 16 indicating a TOP clause

Name	Attribute	Expression	Flag	Name1	Name2
qryTop20SDAUsersAcYear	0		0		
qryTop20SDAUsersAcYear	1		1		
qryTop20SDAUsersAcYear	3		16 20		
qryTop20SDAUsersAcYear	5		WhoseOn		
qryTop20SDAUsersAcYear	6	Count(WhoseOn.LogInID)	0	TotalLogins	
qryTop20SDAUsersAcYear	6	WhoseOn.TeacherID	0		
qryTop20SDAUsersAcYear	8	((((WhoseOn.TimeOn)>=GetAcYearStart()))			
qryTop20SDAUsersAcYear	9	WhoseOn.TeacherID	0		
qryTop20SDAUsersAcYear	10	((((WhoseOn.TeacherID)<>"*****"))			
qryTop20SDAUsersAcYear	11	Count(WhoseOn.LogInID)	D		
qryTop20SDAUsersAcYear	11	WhoseOn.TeacherID			
qryTop20SDAUsersAcYear	255				

Records: 1 of 12
 Filtered
 Search

8. SELECT Query with Ambiguous Joins



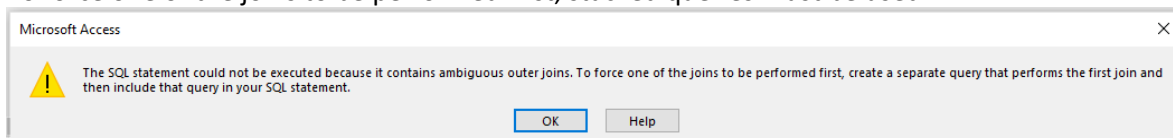
```
SELECT Table1.ID, Table1.DummyTest, Table1.NumberText, Table2.TextField, Table3.TextField2
FROM Table3
LEFT JOIN (Table1 LEFT JOIN Table2 ON Table1.ID = Table2.ID) ON Table3.ID = Table2.ID;
```

Attribute 7 has Flag = 2 for both rows indicating 2 left joins

Name	Attribute	Expression	Flag	Name1	Name2
qryAmbiguousJoins	0		0		
qryAmbiguousJoins	1		1		
qryAmbiguousJoins	3		0		
qryAmbiguousJoins	5			Table3	
qryAmbiguousJoins	5			Table2	
qryAmbiguousJoins	5			Table1	
qryAmbiguousJoins	6	Table3.TextField2	0		
qryAmbiguousJoins	6	Table2.TextField	0		
qryAmbiguousJoins	6	Table1.NumberText	0		
qryAmbiguousJoins	6	Table1.DummyTest	0		
qryAmbiguousJoins	6	Table1.ID	0		
qryAmbiguousJoins	7	Table3.ID = Table2.ID	2	Table3	Table2
qryAmbiguousJoins	7	Table1.ID = Table2.ID	2	Table1	Table2
qryAmbiguousJoins	255				

NOTE:

Queries with ambiguous joins cannot be executed until the join directions are fixed.
To force one of the joins to be performed first, stacked queries must be used



9. Aggregate Query – GROUP BY

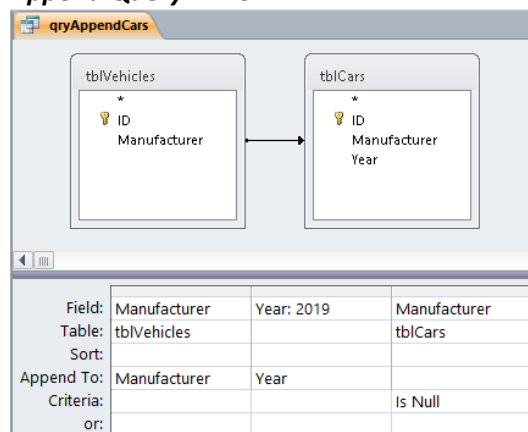
The screenshot shows the design view of a query named 'qryCarColoursCount'. The query is based on the 'qryCars' table. The design grid shows fields: Manufacturer, Year, Model, Colours: Colour, and Price. The criteria row shows '2011' for Year and '<15000' for Price. The total row shows 'Group By' for Manufacturer, Year, and Model, and 'Count' for Colours: Colour.

```
SELECT DISTINCT qryCars.Manufacturer, qryCars.Year, qryCars.Model, Count(qryCars.Colour) AS Colours
FROM qryCars
WHERE (((qryCars.Price)<15000))
GROUP BY qryCars.Manufacturer, qryCars.Year, qryCars.Model
HAVING (((qryCars.Year)=2011))
ORDER BY qryCars.Manufacturer, qryCars.Model;
```

Attribute 10 shows the HAVING clause used to filter the Year field in the aggregate query

Name	Attribute	Flag	Expression	Name1	Name2
qryCarColoursCount	0	0			
qryCarColoursCount	1	1			
qryCarColoursCount	3	2			
qryCarColoursCount	5			qryCars	
qryCarColoursCount	6	0	Count(qryCars.Colour)	Colours	
qryCarColoursCount	6	0	qryCars.Manufacturer		
qryCarColoursCount	6	0	qryCars.Year		
qryCarColoursCount	6	0	qryCars.Model		
qryCarColoursCount	8		((qryCars.Price)<15000))		
qryCarColoursCount	9	0	qryCars.Manufacturer		
qryCarColoursCount	9	0	qryCars.Year		
qryCarColoursCount	9	0	qryCars.Model		
qryCarColoursCount	10		((qryCars.Year)=2011))		
qryCarColoursCount	11		qryCars.Manufacturer		
qryCarColoursCount	11		qryCars.Model		
qryCarColoursCount	255				

10. Append Query – INSERT



```
INSERT INTO tblCars ( Manufacturer, [Year] )
SELECT DISTINCT tblVehicles.Manufacturer, 2019 AS [Year]
FROM tblVehicles LEFT JOIN tblCars ON tblVehicles.Manufacturer = tblCars.Manufacturer
WHERE (((tblCars.Manufacturer) Is Null));
```

Attribute 1 has Flag = 3 indicating an INSERT (Append) query

Name	Attribute	Flag	Expression	Name1	Name2
qryAppendCars	0	0			
qryAppendCars	1	3		tblCars	
qryAppendCars	3	2			
qryAppendCars	5			tblCars	
qryAppendCars	5			tblVehicles	
qryAppendCars	6	0	2019	Year	Year
qryAppendCars	6	0	tblVehicles.Manufacturer		Manufacturer
qryAppendCars	7	2	tblVehicles.Manufacturer = tblCars.Manufacturer	tblVehicles	tblCars
qryAppendCars	8		((tblCars.Manufacturer) Is Null))		
qryAppendCars	255				

11. APPEND Query (INSERT INTO ...VALUES)

SQL view only

```
INSERT INTO tblSettings ( ID, ItemName, ItemValue )
VALUES (9, 'WebPage', 'https://www.isladogs.co.uk/recover-deleted-objects/');
```

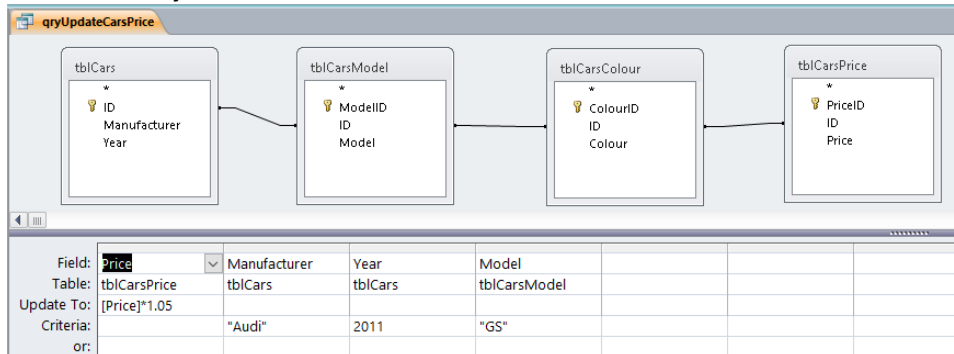
NOTE: If query is saved in Design View, Access changes this to a standard INSERT INTO query

Attribute 1 has Flag = 3 indicating an INSERT (Append) query

Name	Attribute	Expression	Flag	Name1	Name2
qryAppendValues	0		0		
qryAppendValues	1		3	tblSettings	
qryAppendValues	6 9		-32768		ID
qryAppendValues	6 'WebPage'		-32768		ItemName
qryAppendValues	6 'https://www.isladogs.co.uk/recover-deleted-o		-32768		ItemValue
qryAppendValues	255				

Records: 1 of 6

12. UPDATE Query

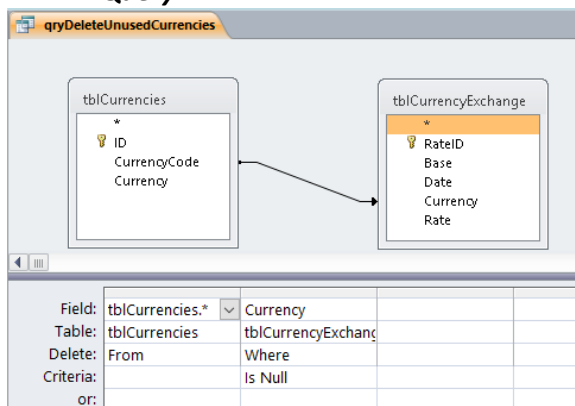


```
UPDATE ((tblCars INNER JOIN tblCarsModel ON tblCars.ID = tblCarsModel.ID) INNER JOIN tblCarsColour ON
tblCarsModel.ID = tblCarsColour.ID) INNER JOIN tblCarsPrice ON tblCarsColour.ID = tblCarsPrice.ID
SET tblCarsPrice.Price = [Price]*1.05
WHERE (((tblCars.Manufacturer)="Audi") AND ((tblCars.Year)=2011) AND ((tblCarsModel.Model)="GS"));
```

Attribute 1 has Flag = 4 indicating an UPDATE query

Name	Attribute	Flag	Expression	Name1	Name2
qryUpdateCarsPrice	0	0			
qryUpdateCarsPrice	1	4			
qryUpdateCarsPrice	3	2			
qryUpdateCarsPrice	5			tblCarsPrice	
qryUpdateCarsPrice	5			tblCarsModel	
qryUpdateCarsPrice	5			tblCarsColour	
qryUpdateCarsPrice	5			tblCars	
qryUpdateCarsPrice	6	0	[Price]*1.05		tblCarsPrice.Price
qryUpdateCarsPrice	7	1	tblCarsColour.ID = tblCarsPrice.ID	tblCarsColour	tblCarsPrice
qryUpdateCarsPrice	7	1	tblCarsModel.ID = tblCarsColour.ID	tblCarsModel	tblCarsColour
qryUpdateCarsPrice	7	1	tblCars.ID = tblCarsModel.ID	tblCars	tblCarsModel
qryUpdateCarsPrice	8		((tblCars.Manufacturer)="Audi") AND ((tblCars.Year)=2011) AND ((tblCarsModel.Model)="GS")		
qryUpdateCarsPrice	255				

13. DELETE Query



```
DELETE tblCurrencies.*, tblCurrencyExchange.Currency
FROM tblCurrencies LEFT JOIN tblCurrencyExchange ON tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency
WHERE (((tblCurrencyExchange.Currency) Is Null));
```

Attribute 1 has Flag = 5 indicating a DELETE query

Name	Attribute	Flag	Expression	Name1	Name2
qryDeleteUnusedCurrencies	0	0			
qryDeleteUnusedCurrencies	1	5			
qryDeleteUnusedCurrencies	3	0			
qryDeleteUnusedCurrencies	5			tblCurrencyExc	
qryDeleteUnusedCurrencies	5			tblCurrencies	
qryDeleteUnusedCurrencies	6	0	tblCurrencyExchange.Currency		
qryDeleteUnusedCurrencies	6	0	tblCurrencies.*		
qryDeleteUnusedCurrencies	7	2	tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency	tblCurrencies	tblCurrencyExchange
qryDeleteUnusedCurrencies	8		((tblCurrencyExchange.Currency) Is Null))		
qryDeleteUnusedCurrencies	255				

14. CROSSTAB Query

qryCarsCrosstab					
qryCars * Manufacturer Year Model Colour Price					
Field:	Manufacturer	Year	Model	Colour	Manufacturer
Table:	qryCars	qryCars	qryCars	qryCars	qryCars
Total:	Group By	Group By	Group By	Count	Where
Crosstab:	Row Heading	Row Heading	Column Heading	Value	
Sort:	Ascending	Ascending			
Criteria:					"BMW" Or "Audi"
or:					

```

TRANSFORM Count(qryCars.Colour) AS CountOfColour
SELECT qryCars.Manufacturer, qryCars.Year
FROM qryCars
WHERE (((qryCars.Manufacturer)="BMW" Or (qryCars.Manufacturer)="Audi"))
GROUP BY qryCars.Manufacturer, qryCars.Year
ORDER BY qryCars.Manufacturer, qryCars.Year
PIVOT qryCars.Model;
  ON tblCurrencies.CurrencyCode = tblCurrencyExchange.Currency
WHERE (((tblCurrencyExchange.Currency) Is Null));

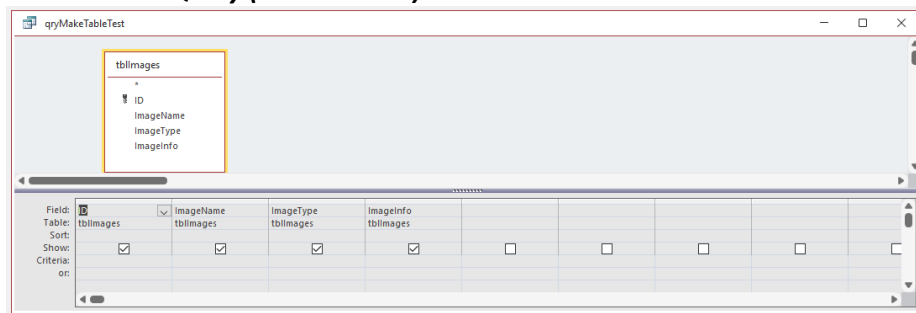
```

Attribute 1 has Flag = 6 indicating a TRANSFORM (Crosstab) query

Name	Attribute	Flag	Expression	Name1	Name2
qryCars_Crosstab	0	0			
qryCars_Crosstab	1	6			
qryCars_Crosstab	3	0			
qryCars_Crosstab	5			qryCars	
qryCars_Crosstab	6	0	Count{qryCars.[Colour]}	CountOfColour	
qryCars_Crosstab	6	1	qryCars.[Model]		
qryCars_Crosstab	6	2	qryCars.[Year]		
qryCars_Crosstab	6	2	qryCars.[Manufacturer]		
qryCars_Crosstab	8		((qryCars.[Manufacturer])="BMW" Or (qryCars.[Manufacturer])="Audi"))		
qryCars_Crosstab	9	1	qryCars.[Model]		
qryCars_Crosstab	9	2	qryCars.[Year]		
qryCars_Crosstab	9	2	qryCars.[Manufacturer]		
qryCars_Crosstab	11		qryCars.[Year]		
qryCars_Crosstab	11		qryCars.[Manufacturer]		
qryCars_Crosstab	255				

New items to add:

15. MAKE TABLE Query (INSERT INTO)



```
SELECT tblImages.ID, tblImages.ImageName, tblImages.ImageType, tblImages.ImageInfo
INTO tblImagesBKP
FROM tblImages;
```

Attribute 1 has Flag = 2 indicating a SELECT ...INTO (Make Table) query

Name	Attribute	Expression	Flag	Name1	Name2
qryMakeTableTest	0		0		
qryMakeTableTest	1		2	tblImagesBKP	
qryMakeTableTest	3		0		
qryMakeTableTest	5		0	tblImages	
qryMakeTableTest	6	tblImages.ID	0		
qryMakeTableTest	6	tblImages.ImageName	0		
qryMakeTableTest	6	tblImages.ImageType	0		
qryMakeTableTest	6	tblImages.ImageInfo	0		
qryMakeTableTest	255				

16. UNION query

SQL view only:

```
SELECT DISTINCT MSysObjectsEXT.Name, MSysObjectsEXT.Type, MSysObjectsEXT.Flags, Hex([MSysObjectsEXT.Flags]) AS
HexFlag, tblSysObjectFlags.Object, tblSysObjectFlags.Category
FROM MSysObjectsEXT INNER JOIN tblSysObjectFlags ON (MSysObjectsEXT.Flags = tblSysObjectFlags.Deleted) AND
(MSysObjectsEXT.Type = tblSysObjectFlags.Type)
WHERE (((MSysObjectsEXT.Name) Like '~TMPCLP*') AND (MSysObjectsEXT.Type) <>-32761)
UNION SELECT DISTINCT MSysObjectsEXT.Name, MSysObjectsEXT.Type, MSysObjectsEXT.Flags,
Hex([MSysObjectsEXT.Flags]) AS HexFlag, tblSysObjectFlags.Object, tblSysObjectFlags.Category
FROM MSysObjectsEXT INNER JOIN tblSysObjectFlags ON (MSysObjectsEXT.Flags = tblSysObjectFlags.HiddenDeleted)
AND (MSysObjectsEXT.Type = tblSysObjectFlags.Type)
WHERE (((MSysObjectsEXT.Name) Like '~TMPCLP*') AND (MSysObjectsEXT.Type) <>-32761)
UNION SELECT DISTINCT MSysObjectsEXT.Name, MSysObjectsEXT.Type, MSysObjectsEXT.Flags,
Hex([MSysObjectsEXT.Flags]) AS HexFlag, tblSysObjectFlags.Object, GetModuleType(MSysObjectsEXT.Name) AS
Category
FROM MSysObjectsEXT INNER JOIN tblSysObjectFlags ON (MSysObjectsEXT.Flags = tblSysObjectFlags.HiddenDeleted)
AND (MSysObjectsEXT.Type = tblSysObjectFlags.Type)
WHERE (((MSysObjectsEXT.Name) Like '~TMPCLP*') AND (MSysObjectsEXT.Type) =-32761);
```

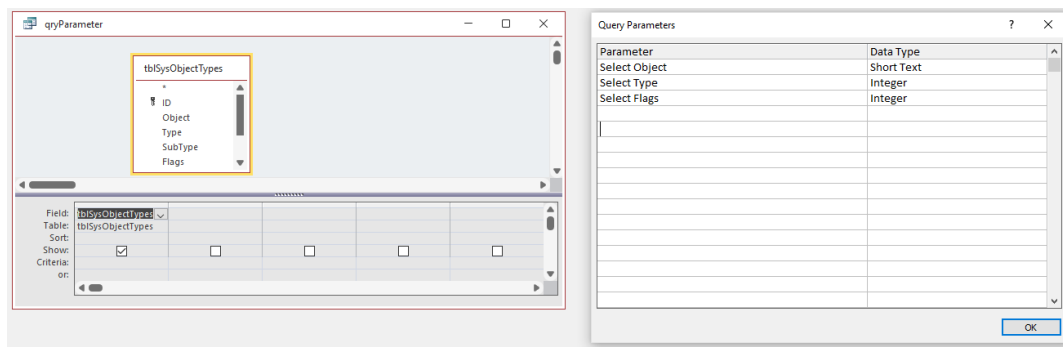
Attribute 1 has Flag = 9 indicating a UNION query

Attribute 5 uses internal identifiers for each UNION segment in the Name2 field

Name	Attribute	Expression	Flag	Name1	Name2
qryMSysObjectsTMPCLP	0		0		
qryMSysObjectsTMPCLP	1		9		
qryMSysObjectsTMPCLP	3		3		
qryMSysObjectsTMPCLP	5	SELECT DISTINCT MSysObjectsEXT.Name, MSysObjectsEXT.Type, MSysObj			X7YZ__1
qryMSysObjectsTMPCLP	5	SELECT DISTINCT MSysObjectsEXT.Name, MSysObjectsEXT.Type, MSysObj			X7YZ__2
qryMSysObjectsTMPCLP	255				

17. PARAMETER Query

Filter parameters can also be entered using the Query Parameters dialog in the Query Design ribbon

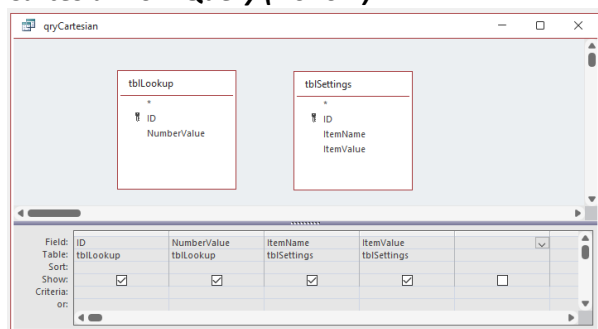


PARAMETERS [Select Object] Text (255), [Select Type] Short, [Select Flags] Short;
 SELECT tblSysObjectTypes.*
 FROM tblSysObjectTypes;

Attribute 2 has the PARAMETER info where Flag 3 is INTEGER datatype & 10 is TEXT

Name	Attribute	Expression	Flag	Name1	Name2
qryParameter	0		0		
qryParameter	1		1		
qryParameter	2		10	Select Object	
qryParameter	2		3	Select Type	
qryParameter	2		3	Select Flags	
qryParameter	3		0		
qryParameter	5			tblSysObjectTypes	
qryParameter	6	tblSysObjectTypes.*	0		
qryParameter	255				

18. Cartesian Join Query (No Join)

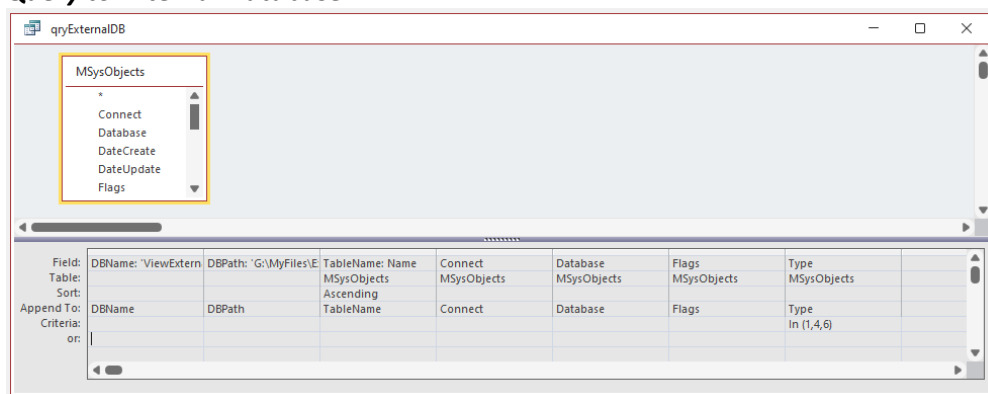


SELECT tblLookup.ID, tblLookup.NumberValue, tblSettings.ItemName, tblSettings.ItemValue
 FROM tblLookup, tblSettings;

Attribute 7 is missing as there are no joins

Name	Attribute	Expression	Flag	Name1	Name2
qryCartesian	0		0		
qryCartesian	1		1		
qryCartesian	3		0		
qryCartesian	5			tblLookup	
qryCartesian	5			tblSettings	
qryCartesian	6	tblLookup.ID	0		
qryCartesian	6	tblLookup.NumberValue	0		
qryCartesian	6	tblSettings.ItemName	0		
qryCartesian	6	tblSettings.ItemValue	0		
qryCartesian	255				

19. Query to External Database

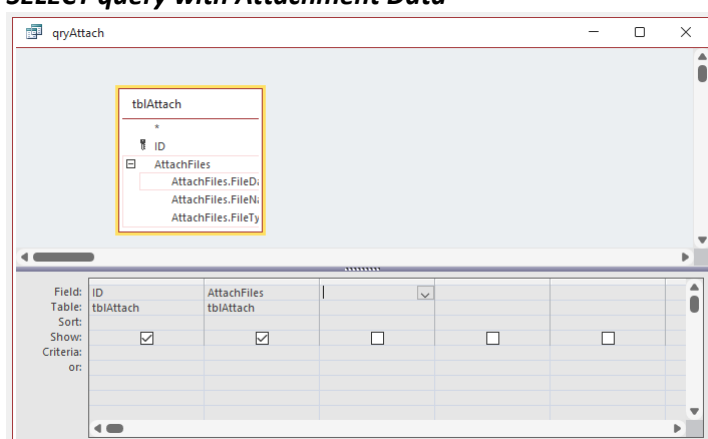


```
INSERT INTO tblTableList ( DBName, DBPath, TableName, [Connect], [Database], Flags, Type )
SELECT 'ViewExternalTables_v2.7 Pro.accdb' AS DBName,
'G:\MyFiles\ExampleDatabases\ExternalTableViewer\ViewExternalTables_v2.7 Pro.accdb' AS DBPath,
MSysObjects.Name AS TableName, MSysObjects.[Connect], MSysObjects.[Database], MSysObjects.Flags,
MSysObjects.Type FROM MSysObjects IN " [MS
Access;PWD=;DATABASE=G:\MyFiles\ExampleDatabases\ExternalTableViewer\ViewExternalTables_v2.7 Pro.accdb]
WHERE ((MSysObjects.Type) IN (1,4,6)) ORDER BY MSysObjects.Name;
```

Attribute 4 shows the connection string to the external database

Name	Attribute	Expression	Flag	Name1	Name2
qryExternalDB	0		0		
qryExternalDB	1		3	tblTableList	
qryExternalDB	3		0		
qryExternalDB	4	MS Access;PWD=;DATABASE=G:\MyFiles\Examp			
qryExternalDB	5			MSysObjects	
qryExternalDB	6	'ViewExternalTables_v2.7 Pro.accdb'	0	DBName	DBName
qryExternalDB	6	'G:\MyFiles\ExampleDatabases\ExternalTableVie	0	DBPath	DBPath
qryExternalDB	6	MSysObjects.Name	0	TableName	TableName
qryExternalDB	6	MSysObjects.Connect	0		Connect
qryExternalDB	6	MSysObjects.Database	0		Database
qryExternalDB	6	MSysObjects.Flags	0		Flags
qryExternalDB	6	MSysObjects.Type	0		Type
qryExternalDB	8	((MSysObjects.Type) In (1,4,6))			
qryExternalDB	11	MSysObjects.Name			
qryExternalDB	255				

20. SELECT query with Attachment Data



```
SELECT tblAttach.ID, tblAttach.AttachFiles
FROM tblAttach;
```

Attribute 12 Flag =2 indicates the query includes an attachment field

Name	Attribute	Expression	Flag	Name1	Name2
qryAttach	0		0		
qryAttach	1		1		
qryAttach	3		0		
qryAttach	5		tblAttach		
qryAttach	6	tblAttach.ID	0		
qryAttach	6	tblAttach.AttachFiles	0		
qryAttach	12		2		
qryAttach	255				

The Attachment data can be expanded:

tblAttach					
* ID					
AttachFiles					
AttachFiles.FileData					
AttachFiles.FileName					
AttachFiles.FileType					

Field:	ID	AttachFiles	AttachFiles.FileData	AttachFiles.FileName	AttachFiles.FileType
Table:	tblAttach	tblAttach	tblAttach	tblAttach	tblAttach
Sort:					
Show:	☒	☒	☒	☒	☒
Criteria:					
or:					

*SELECT tblAttach.ID, tblAttach.AttachFiles, tblAttach.AttachFiles.FileData, tblAttach.AttachFiles.FileName, tblAttach.AttachFiles.FileType
FROM tblAttach;*

Attribute 12 Name1 field shows the deep hidden attached table info

Name	Attribute	Expression	Flag	Name1	Name2
qryAttachFull	0		0		
qryAttachFull	1		1		
qryAttachFull	3		0		
qryAttachFull	5		tblAttach		
qryAttachFull	6	tblAttach.ID	0		
qryAttachFull	6	tblAttach.AttachFiles	0		
qryAttachFull	6	tblAttach.AttachFiles.FileData	0		
qryAttachFull	6	tblAttach.AttachFiles.FileName	0		
qryAttachFull	6	tblAttach.AttachFiles.FileType	0		
qryAttachFull	12		2	qryAttachFull_6EB2D4686FF847C0B4DDAED57295DA4A	
qryAttachFull	255				

21. SELECT Query with MVF data

Similar to Attachment field – example below includes the attached MVF data

tblMVF2			
* ID			
MVF2			
MVF2.Value			

Field:	ID	MVF2	MVF2.Value
Table:	tblMVF2	tblMVF2	tblMVF2
Sort:			
Show:	☒	☒	☒
Criteria:			
or:			

```
SELECT tblIMVF2.ID, tblIMVF2.MVF2, tblIMVF2.MVF2.Value
FROM tblIMVF2;
```

Once again, Attribute 12 has Flag = 2 and the deep hidden attached table info is shown in the Name1 field

Name	Attribute	Expression	Flag	Name1	Name2
qryMVF2	0		0		
qryMVF2	1		1		
qryMVF2	3		0		
qryMVF2	5			tblIMVF2	
qryMVF2	6	tblIMVF2.ID	0		
qryMVF2	6	tblIMVF2.MVF2	0		
qryMVF2	6	tblIMVF2.MVF2.Value	0		
qryMVF2	12		2	qryMVF2_B577C11D5825473786F71E7B28A3F4EE	
qryMVF2	255				

22. SELECT query with Version History Data

This is the third type of complex data field – memo field version history

tblColumnHistory	
ID	MemoField

Field:	ID	MemoField					
Table:	tblColumnHistory	tblColumnHistory					
Sort:							
Show:	☒	☒	☐	☐	☐	☐	☐
Criteria:							
or:							

```
SELECT tblColumnHistory.ID, tblColumnHistory.MemoField
FROM tblColumnHistory;
```

Attribute 12 has Flag=1 for this complex datatype

Name	Attribute	Expression	Flag	Name1	Name2
qryColumnHistory	0		0		
qryColumnHistory	5			tblColumnHistory	
qryColumnHistory	6	tblColumnHistory.ID	0		
qryColumnHistory	6	tblColumnHistory.MemoField	0		
qryColumnHistory	12		1		
qryColumnHistory	255				

23. SELECT Query with non equi-join & table aliases

In this example, 2 copies of the same table are joined (self-join) with one field connected with a non equal join (A.Reading > B.ReadingDate)

NOTE:

This method is commonly used to get a value from the previous record.

In this case, the previous meter reading

Aliases A/B have been used for each copy of the table

Non-equi-joins cannot be displayed in design view

```
SELECT A.MeterFK, A.ReadingDate, A.Reading, B.ReadingDate, B.Reading, [a].[readingdate]-[b].[readingdate] AS days,
[A].[reading]-[b].[reading]-(100000*([a].[readingdate]-#11/12/2020#)) AS used, [used]/[days] AS perday
```

FROM tblMeterReadings AS A INNER JOIN tblMeterReadings AS B ON (A.Reading > B.ReadingDate) AND (A.MeterFK = B.MeterFK)
WHERE (((A.MeterFK)<=1) AND ((B.Estimate)=False) AND ((A.Estimate)=False))
ORDER BY A.MeterFK, A.ReadingDate;

Attribute 7 shows all joins correctly including the non equi-join

Attribute 5 Name1 & Name2 fields show the table names & aliases respectively

Name	Attribute	Flag	Expression	Name1	Name2
qryMeterReadingsNonEquiJoin	0	0			
qryMeterReadingsNonEquiJoin	5			tblMeterReadings	A
qryMeterReadingsNonEquiJoin	5			tblMeterReadings	B
qryMeterReadingsNonEquiJoin	6	0	[A].[readingdate]-[B].[readingdate]/(1000000*([A].[readingdate]-#11/1 used		
qryMeterReadingsNonEquiJoin	6	0	A.MeterFK		
qryMeterReadingsNonEquiJoin	6	0	A.ReadingDate		
qryMeterReadingsNonEquiJoin	6	0	A.Reading		
qryMeterReadingsNonEquiJoin	6	0	B.ReadingDate		
qryMeterReadingsNonEquiJoin	6	0	[a].[readingdate]-[b].[readingdate]	days	
qryMeterReadingsNonEquiJoin	6	0	[used]/[days]	perday	
qryMeterReadingsNonEquiJoin	6	0	B.Reading		
qryMeterReadingsNonEquiJoin	7	1	A.Reading > B.ReadingDate	A	B
qryMeterReadingsNonEquiJoin	7	1	A.MeterFK = B.MeterFK	A	B
qryMeterReadingsNonEquiJoin	8		(((A.MeterFK)<=1) AND ((B.Estimate)=False) AND ((A.Estimate)=False))		
qryMeterReadingsNonEquiJoin	11		A.MeterFK		
qryMeterReadingsNonEquiJoin	11		A.ReadingDate		
qryMeterReadingsNonEquiJoin	255				

NOTE:

In this example, note that Attribute 1 (SELECT) and 3 are both missing
I will explain this point in the second part of this article (to follow)

24. Data Definition Query

SQL view only

ALTER TABLE tblAttach ADD COLUMN T TEXT(5);

Attribute 1 has Flag = 7 indicating a data definition (DDL) query

Name	Attribute	Expression	Flag	Name1	Name2
qryDDLTest	0		0		
qryDDLTest	1	ALTER TABLE tblAttach ADD COLUMN T TEXT(5);	7		
qryDDLTest	255				

25. Passthrough Query

SQL view only

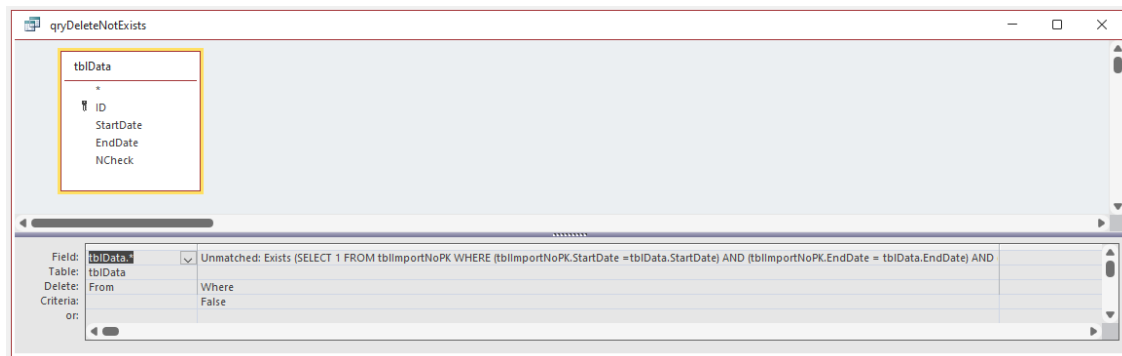
EXEC spGetStudentAttendanceMarks 12876

Attribute 1 has Flag = 8 indicating a passthrough query

Name	Attribute	Expression	Flag	Name1	Name2
qryGetLessonAttendance	0		0		
qryGetLessonAttendance	1	EXEC spGetStudentAttendanceMarks 12876	8	ODBC;DRIVER=SQL Server;SERVE	
qryGetLessonAttendance	255				

26. Subquery

In this example, a DELETE query uses a subquery to only delete records that do not match certain criteria.
To do this, the query checks if the subquery criteria are False (Not Exists)



DELETE tblData.*; **Exists** (SELECT 1 FROM tblImportNoPK WHERE (tblImportNoPK.StartDate =tblData.StartDate) AND (tblImportNoPK.EndDate = tblData.EndDate) AND (tblImportNoPK.NCheck = tblData.NCheck)) AS Unmatched
FROM tblData
WHERE (((Exists (SELECT 1 FROM tblImportNoPK WHERE (tblImportNoPK.StartDate =tblData.StartDate) AND (tblImportNoPK.EndDate = tblData.EndDate) AND (tblImportNoPK.NCheck = tblData.NCheck)))=False));

The subquery info is shown in attribute 6 (Field info) and attribute 8 (WHERE clause)

Name	Attribute	Expression	Flag	Name1	Name2
qryDeleteNotExists	0		0		
qryDeleteNotExists	1		5		
qryDeleteNotExists	5			tblData	
qryDeleteNotExists	6	Exists (SELECT 1 FROM tblImportNoPK WHERE (tblImportNoPK.StartDate	0	Unmatched	
qryDeleteNotExists	6	tblData.*	0		
qryDeleteNotExists	8	((Exists (SELECT 1 FROM tblImportNoPK WHERE (tblImportNoPK.StartDa			
qryDeleteNotExists	255				

27. TEMP query (form/report record source)

Access creates 'temp' queries for use as form/ report record sources and for combo / listboxes row sources. These always begin with **~sq_ ...** and are not visible in the navigation pane. However, they can be viewed from the property sheet for the object or control

fsusbClassList : Query Builder									
ClassRecords - Tag - ClassRecordID - PupilID - UPN - ClassID					PupilData - PupilID - Surname - Forename - LegalSurname - LegalForename				
Field:	PupilID	Student: [Surname] &	TGp: [YearGroup] & T	ClassID	Gender	COPLevel	Photo Available	Surname	Forename
Table:	PupilData	PupilData	PupilData	ClassRecords	PupilData	PupilData	PupilData	PupilData	PupilData
Total:	Group By	Group By	Group By	Group By	Group By	Group By	Group By	Group By	Group By
Sort:								Ascending	Ascending
Show:	☒	☒	☒	☒	☒	☒	☒	☒	☒
Criteria:									
or:									

SELECT DISTINCTROW PupilData.PupilID, [Surname] & " " & [Forename] AS Student, [YearGroup] & [TutorGroup] AS TGp, ClassRecords.ClassID, PupilData.Gender, PupilData.COPLevel, PupilData.[Photo Available], PupilData.Surname, PupilData.Forename
FROM ClassRecords INNER JOIN PupilData ON ClassRecords.PupilID = PupilData.PupilID
GROUP BY PupilData.PupilID, [Surname] & " " & [Forename], [YearGroup] & [TutorGroup], ClassRecords.ClassID, PupilData.Gender, PupilData.COPLevel, PupilData.[Photo Available], PupilData.Surname, PupilData.Forename
ORDER BY PupilData.Surname, PupilData.Forename;

Attribute 3 Flag = 9 indicates a temp query. In this case, it is used as a form record source
 Attribute 2 Flag = 0 indicating the value of the filtered field used in the record source

Double underscores are used (__ChildID) to identify the master field used in the master/child field join

Name	Attribute	Expression	Flag	Name1	Name2
~sq_cfrmClassList~sq_cfsbClassList	0		0		
~sq_cfrmClassList~sq_cfsbClassList	2		0	__ClassID	
~sq_cfrmClassList~sq_cfsbClassList	3		9		
~sq_cfrmClassList~sq_cfsbClassList	5	SELECT DISTINCTROW PupilData.PupilID, [Surname]			frmClassList
~sq_cfrmClassList~sq_cfsbClassList	8	([_ClassID] = ClassID)			
~sq_cfrmClassList~sq_cfsbClassList	255				

28. Deleted Query

Recently deleted queries are renamed with a ~TMPCLP prefix and are no longer visible in the navigation pane. However, their properties can be determined as for all other queries.

Attribute values are identical to those for the same query before it was deleted.

Name	Attribute	Expression	Flag	Name1	Name2
~TMPCLP630531	0		0		
~TMPCLP630531	5			tblTableList	
~TMPCLP630531	5			tblTableTypes	
~TMPCLP630531	6	tblTableList.DBName	0		
~TMPCLP630531	6	tblTableList.DBPath	0		
~TMPCLP630531	6	tblTableList.TableName	0		
~TMPCLP630531	6	tblTableList.TableType	0		
~TMPCLP630531	6	tblTableList.Type	0		
~TMPCLP630531	6	tblTableList.Flags	0		
~TMPCLP630531	6	tblTableTypes.Hidden	0		
~TMPCLP630531	6	tblTableTypes.System	0		
~TMPCLP630531	6	tblTableTypes.Linked	0		
~TMPCLP630531	6	tblTableTypes.DeepHidden	0		
~TMPCLP630531	6	tblTableList.Deleted	0		
~TMPCLP630531	6	tblTableTypes.Notes	0		
~TMPCLP630531	7	tblTableList.Flags = tblTableTypes.Flags	2	tblTableList	tblTableTypes
~TMPCLP630531	7	tblTableList.Type = tblTableTypes.Type	2	tblTableList	tblTableTypes
~TMPCLP630531	8	((tblTableList.DBPath)=[Forms]![frmClassList])			
~TMPCLP630531	255				

So, in theory, you can reconstruct the query from this information if it is still needed.

However, luckily there is an easier method:

See <https://www.isladogs.co.uk/recover-deleted-objects/> **Recover Deleted Database Objects**

NOTE: the ~TMPCLP query is deleted permanently when the database is compacted or closed.

29. Summary

The contents of the **MSysQueries** table can be used to view the structure of all saved queries. All queries, no matter how complex, work in exactly the same way as those listed above.

In principle, it is possible to 'reverse engineer' the **MSysQueries** table data to construct **query SQL** based on the rules above. Thankfully, there is normally no need as **Access** does the hard work instead!

30. Downloads

Click to download:

This article as a PDF file: [MSysQueries Info](#)

The example database: [MSysQueries Example](#)

Further reading:

SO article [What does the data in MSysQueries Mean?](#)

MS article [DataType enumerations \(DAO\)](#)

When a query is saved, the view used during the save is stored and used next time the query is opened.

The second part of this article (available soon) will explain how Access retrieves that design view/SQL view information for future use.

Part 2 to follow ...

This will explain how Access determines whether a query opens in design view or SQL view