Synchronise a table with external data – Part 2

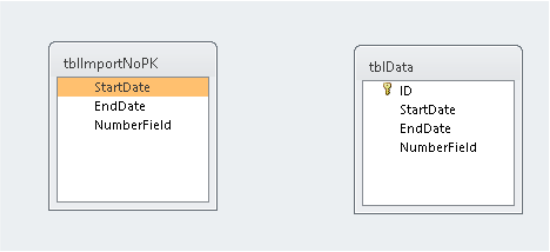
The first part of this article discussed four ways of synchronising data where the external data source includes a primary key field. This is the likely situation where you are synchronising records with another Access table or external database

The second part looks at ways of synchronising data where there is no primary key field in the import table. This is often the case when you are importing data from an Excel or CSV file

As with most processes in Access, there are several ways of achieving this result. Whichever method is used it is recommended that a backup is made before synchronising with external data in case any problems arise.

The following code assumes that:

- the records in tblData are being updated from tblImportNoPK
- both tables have the same fields apart from there being no autonumber PK field in the import table.
- tblImport contains ALL records that should be imported to tblData



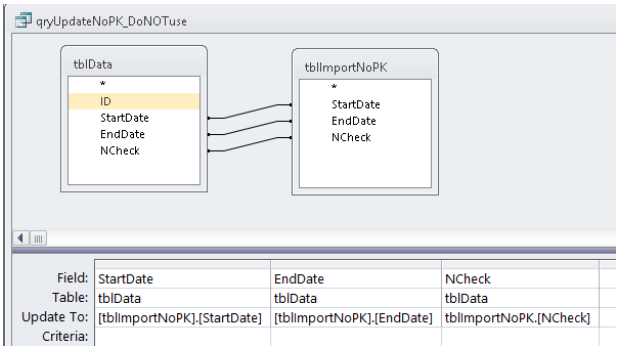
This requires some changes to the code used previously.

More care is needed in checking the outcomes will be what you want before synchronising the data.
This is especially important if you have any NULL values in import or destination tables as a NULL value is not equal to anything else – NOT even another NULL value.

---

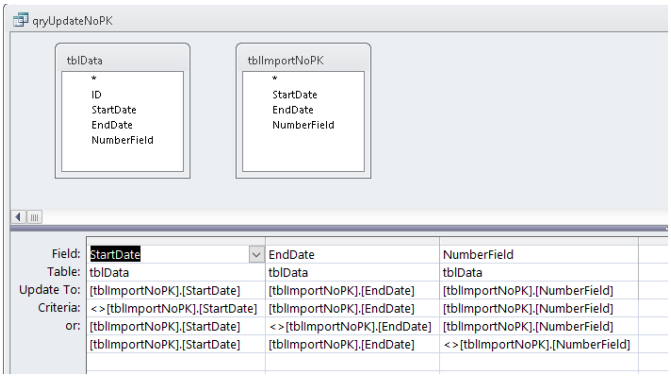
5. Update Existing / Delete Old (using JOIN) / Append New

First update existing records.
The following query won't work as it will only update records if they are the same in both tables
This is of course totally pointless *(qryUpdateNoPK_DoNOTuse)*



DO NOT USE THIS

Instead create a cartesian join query (unlinked tables) and specify conditions to use.
For example, this will update any records where any ONE field has been changed *(qryUpdateNoPK)*



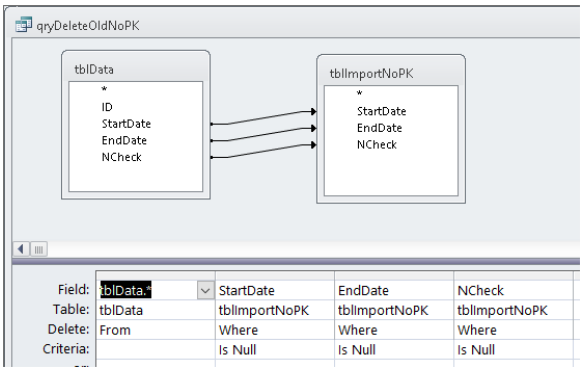
As no PK is involved, you should set Unique Records = Yes

UPDATE DISTINCTROW tblData, tblImportNoPK SET tblData.StartDate = [tblImportNoPK].[StartDate], tblData.EndDate = [tblImportNoPK].[EndDate], tblData.NCheck = [tblImportNoPK].[NCheck]
WHERE (((tblData.StartDate)<>tblImportNoPK.StartDate) And ((tblData.EndDate)=tblImportNoPK.EndDate) And ((tblData.NCheck)=tblImportNoPK.NCheck)) Or (((tblData.StartDate)=tblImportNoPK.StartDate) And ((tblData.EndDate)<>tblImportNoPK.EndDate) And ((tblData.NCheck)=tblImportNoPK.NCheck)) Or (((tblData.StartDate)=tblImportNoPK.StartDate) And ((tblData.EndDate)=tblImportNoPK.EndDate) And ((tblData.NCheck)<>tblImportNoPK.NCheck));

If you have a large number of existing records to check and update, this method will be VERY slow especially if there are a large number of null values (as in the example database)

It will also not pick up any records where TWO or more fields have been changed
Additional criteria would be needed to manage those situations.
The query could get very complicated in such cases. If so, this method is best avoided.

Assuming the method is feasible in your situation, next delete any old records that aren't in the import table – again set Unique Records = Yes *(qryDeleteOldNoPK)*



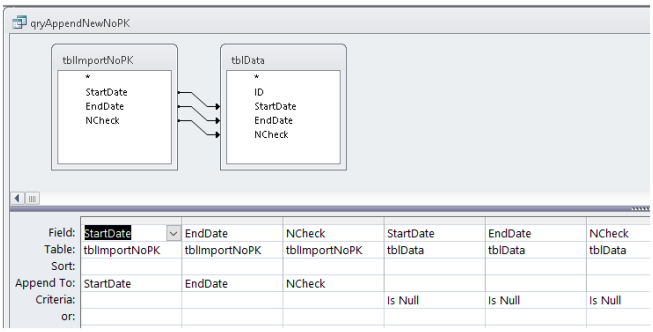
DELETE DISTINCTROW tblData., tblImportNoPK.StartDate, tblImportNoPK.EndDate, tblImportNoPK.NCheck*
FROM tblData LEFT JOIN tblImportNoPK ON (tblData.NCheck = tblImportNoPK.NCheck) AND (tblData.EndDate = tblImportNoPK.EndDate) AND (tblData.StartDate = tblImportNoPK.StartDate)
WHERE (((tblImportNoPK.StartDate) Is Null) AND ((tblImportNoPK.EndDate) Is Null) AND ((tblImportNoPK.NCheck) Is Null));

NOTE:
Once again, this may be very slow if you have a very large number of existing records to check

It may also DELETE records that should be retained if you have duplicate values or NULL values in more than one import fields or destination fields.

If so, edit the query to manage the issue or use a different approach

Finally append all new records. Set **Unique Values = Yes (DISTINCT)** ***(qryAppendNewNoPK)***




```
INSERT INTO tblData ( StartDate, EndDate, NCheck )
SELECT DISTINCT tblImportNoPK.StartDate, tblImportNoPK.EndDate, tblImportNoPK.NCheck
FROM tblImportNoPK LEFT JOIN tblData ON (tblImportNoPK.StartDate = tblData.StartDate) AND
(tblImportNoPK.EndDate = tblData.EndDate) AND (tblImportNoPK.[NCheck] = tblData.[NCheck])
WHERE (((tblData.StartDate) Is Null) AND ((tblData.EndDate) Is Null) AND ((tblData.NCheck) Is Null));
```

NOTE :
A similar warning to the UPDATE & DELETE parts of this method.
The above query will cause DUPLICATE records if you have NULL values in more than one import fields or destination fields. You may be able to edit the query to manage the issue

However, it may be safest to manage null values by first setting them to valid but unused values in each field of the import & destination fields
For example, update null dates to 01/01/9999 and number fields to an unlikely value such as -1000000
You will need to do this for each field separately in the two tables. For example:

```
UPDATE tblData SET tblData.StartDate = #1/1/9999#
WHERE (((tblData.StartDate) Is Null));
```

Once you have done this, run the above queries. After completion run a further update query to revert the modified values in the destination table back to null.
Once again do this for each field separately. For example:

```
UPDATE tblData SET tblData.StartDate = Null
WHERE (((tblData.StartDate)=#1/1/9999#));
```

Overall, this method is much more complex to administer particularly if you have duplicate or null values to manage. It is also likely to be VERY slow particularly for large datasets

You should also be aware that as some records will be deleted, there will be gaps in the autonumber field

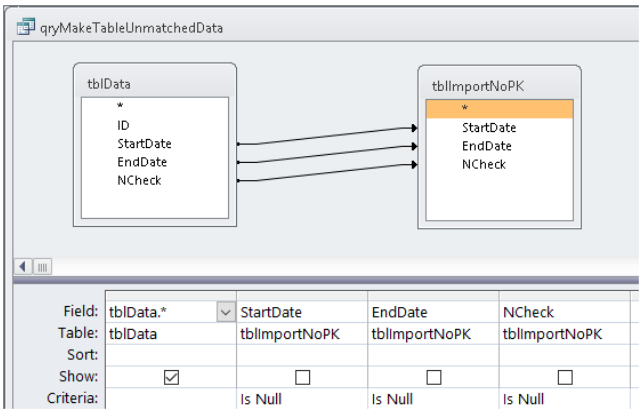
6. Update Existing / Delete Old (TEMP table) / Append New

This is similarto the previous approach but uses a different method for identifying old records for deletion

First of all, update existing records as in method 5 (qryUpdateNoPK)

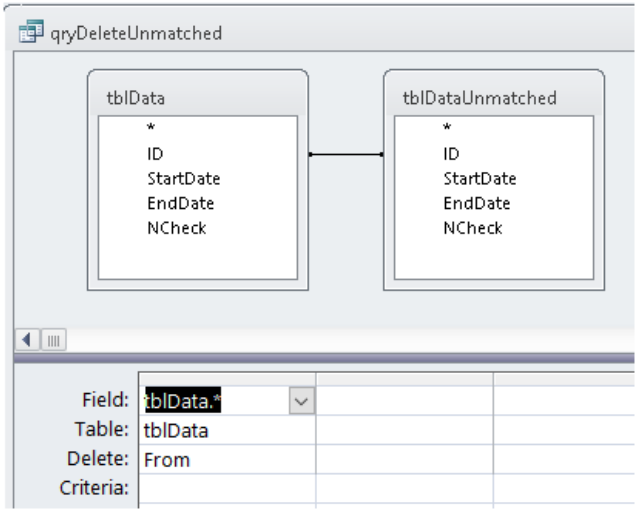
```
UPDATE DISTINCTROW tblData, tblImportNoPK SET tblData.StartDate = [tblImportNoPK].[StartDate],
tblData.EndDate = [tblImportNoPK].[EndDate], tblData.NCheck = [tblImportNoPK].[NCheck]
WHERE (((tblData.StartDate)<>tblImportNoPK.StartDate) And ((tblData.EndDate)=tblImportNoPK.EndDate) And
((tblData.NCheck)=tblImportNoPK.NCheck)) Or (((tblData.StartDate)=tblImportNoPK.StartDate) And
((tblData.EndDate)<>tblImportNoPK.EndDate) And ((tblData.NCheck)=tblImportNoPK.NCheck)) Or
(((tblData.StartDate)=tblImportNoPK.StartDate) And ((tblData.EndDate)=tblImportNoPK.EndDate) And
((tblData.NCheck)<>tblImportNoPK.NCheck));
```

Next create a temp table to identify unmatched records in tblData but not tblImportNoPK
(qryMakeTableUnmatchedData)




```
SELECT tblData.* INTO tblDataUnmatched
FROM tblImportNoPK RIGHT JOIN tblData ON (tblImportNoPK.StartDate = tblData.StartDate) AND
(tblImportNoPK.EndDate = tblData.EndDate) AND (tblImportNoPK.[NCheck] = tblData.[NCheck])
WHERE (((tblImportNoPK.StartDate) Is Null) AND ((tblImportNoPK.EndDate) Is Null) AND ((tblImportNoPK.NCheck)
Is Null));
```

Next delete these unmatched records using the temp table *(qryDeleteUnmatched)*




```
DELETE tblData.*
FROM tblData INNER JOIN tblDataUnmatched ON tblData.ID = tblDataUnmatched.ID;
```

The temp table can now be deleted

```
DoCmd.DeleteObject acTable, "tblDataUnmatched"
```

NOTE: If this method is to be used repeatedly, it is better to keep the temp table and just empty it after use

Finally append all new records setting **Unique Values = Yes** as in method 5 *(qryAppendNewNoPK)*

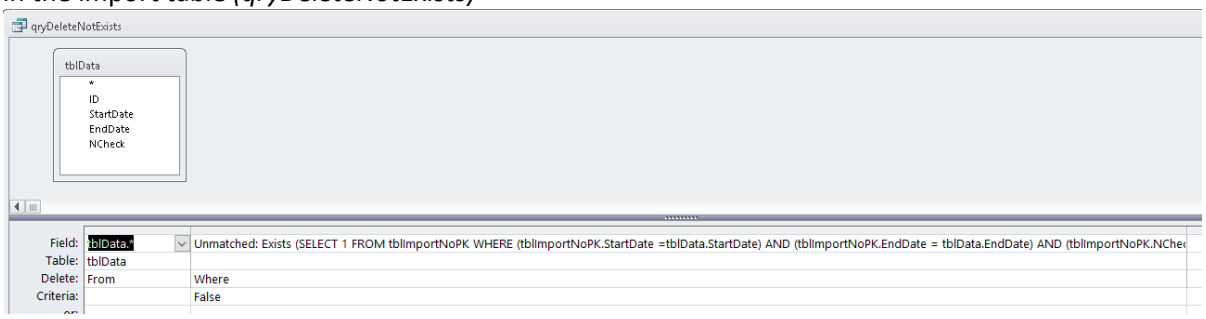
```
INSERT INTO tblData ( StartDate, EndDate, NCheck )
SELECT DISTINCT tblImportNoPK.StartDate, tblImportNoPK.EndDate, tblImportNoPK.NCheck
FROM tblImportNoPK LEFT JOIN tblData ON (tblImportNoPK.StartDate = tblData.StartDate) AND
(tblImportNoPK.EndDate = tblData.EndDate) AND (tblImportNoPK.[NCheck] = tblData.[NCheck])
WHERE (((tblData.StartDate) Is Null) AND ((tblData.EndDate) Is Null) AND ((tblData.NCheck) Is Null));
```

This approach may be slightly easier to manage than method 5 but the total time required will be similar
Once again, as some records will be deleted, there will be gaps in the autonumber field

7. Update Existing / Delete Old (NOT exists) / Append New

This method uses the same update ***(qryUpdateNoPK)*** and append queries ***(qryAppendNewNoPK)*** as in methods 5 & 6

The delete part uses a different approach to identify and then delete records from tblData that do NOT exist in the import table *(qryDeleteNotExists)*




```
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));
```

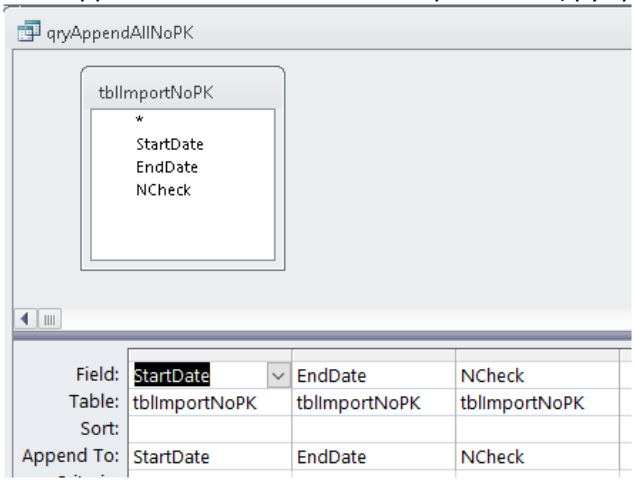
This approach to the delete query is more difficult to design but is often more efficient for Access to process. It may therefore run SLIGHTLY faster than methods 5 or 6

8. Delete All / Append All

This is identical to method 3 above
First delete ALL records *(qryDeleteAll)*

```
DELETE tblData.* FROM tblData;
```

Now append all records from the import table *(qryAppendAllNoPK)*




```
INSERT INTO tblData ( StartDate, EndDate, NCheck )
SELECT tblImportNoPK.StartDate, tblImportNoPK.EndDate, tblImportNoPK.NCheck FROM tblImportNoPK;
```

This is the simplest and most reliable method if the import table contains all the required records as you don't need to manage issues with null values.

It will run MUCH faster than methods 5, 6 or 7.

For most situations, the significant reduction in time more than offsets the additional increase in file size caused by replacing all records

9. Make Table / Append All

This is almost identical to method 4 in the first part of this article

First create a new tblData with PK field using code or a data definition query *(qryCreateDataTable)*

```
CREATE TABLE tblData (ID AUTOINCREMENT NOT NULL PRIMARY KEY, StartDate DATETIME,
EndDate DATETIME, NCheck INT);
```

If using the query designer, this can only be done in SQL view

Next populate the table by appending all records as in method 8 *(qryAppendAllNoPK)*

```
INSERT INTO tblData ( StartDate, EndDate, NCheck )
SELECT tblImportNoPK.StartDate, tblImportNoPK.EndDate, tblImportNoPK.NCheck FROM tblImportNoPK;
```

Once again this is very simple to administer if all required records exist in the import table
It will again run very fast with times similar to method 8 but a larger file 'bloat' due to deletion and creation of tables

10. Combined Upend (AKA Upsert) / Delete Old

This method cannot be used as there is no PK field in the import table

SUMMARY

In cases where there is no PK field in the import table, your choices may be more limited.
Take great care to ensure the results are what you want before synchronising data
Use a SELECT query to check the records before running a DELETE, APPEND or UPDATE query

For comparison, here are the results I obtained for all the tests in both parts of this article:
Tests 1->4 : Import table with PK
All methods are fast but the UPEND query is slightly slower
Tests 5->9 : Import table with no PK
Methods 5-7 are MUCH slower and more complex to design
Methods 8 & 9 are about as fast as the first 4 methods

Average Results

Workstation	Test Count	Test Type	Average Time (s)	Average File Size Change (KB)
COLIN-PC	2	1. Update Existing /Append New / Delete Old	0.66	710
COLIN-PC	2	2. Combined Upend / Delete Old	1.09	752
COLIN-PC	2	3. Delete All / Append All	0.73	770
COLIN-PC	2	4. Make Table / Append All	0.59	724
COLIN-PC	2	5. Update Existing / Delete Old (JOIN) / Append New	20.45	722
COLIN-PC	2	6. Update Existing / Delete Old (TEMP table) / Append New	20.55	752
COLIN-PC	2	7. Update Existing / Delete Old (NOT EXISTS) / Append New	20.41	732
COLIN-PC	2	8. Delete All / Append All	0.61	764
COLIN-PC	2	9. Make Table / Append All	0.60	740

Record: 1 of 9 | No Filter | Search

NOTE:

a) Compacting will recover much of the space added after deleting records
b) Repeatedly creating / overwriting tables can cause instability & in some cases may lead to corruption
c) For one-off synchronisation, it may be sufficient to create the queries you need and discard them after use
However, if you will need to repeat the operation, it is sensible to create procedures to run each of the query definitions in turn
d) The example database **SyncDataExample.accdb** contains all the tables / queries and procedures used in this article