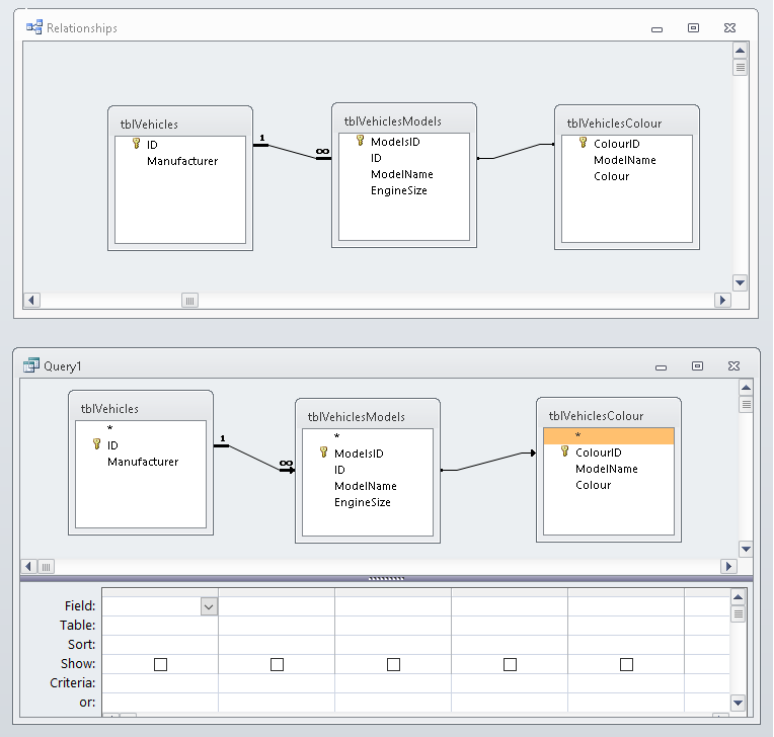
A bit of Relationships advice – Part 1

Many new users of Access are unsure about the difference between query joins and table relationships.

When a query is created, tables (and queries) can be joined in different ways (inner/left/right joins) whether or not a relationship has been defined at table level. See the article Types of Query Joins for more information Furthermore, different query joins can be used with any table relationships already created.



In truth, there are many similarities between query joins and table relationships.
If relationships have been applied, these will automatically appear for those tables when used in the query designer window. However, that isn't important enough to justify their use

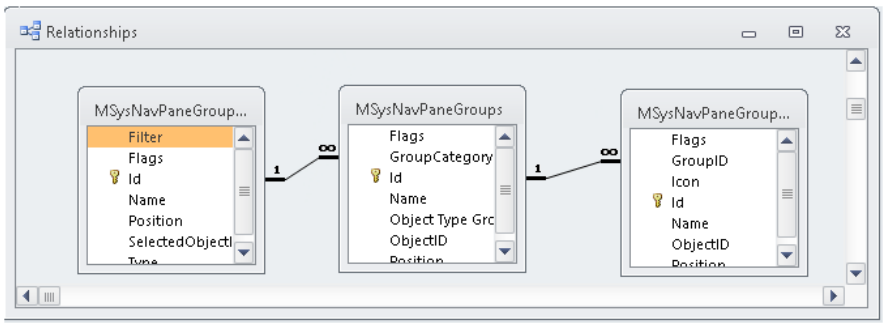
As a result, there is some disagreement amongst developers about the use of table relationships
Some developers apply them rigorously to any tables with linked data. Others hardly ever use them.

However, relationships have another very important purpose – enforcing **referential integrity**.

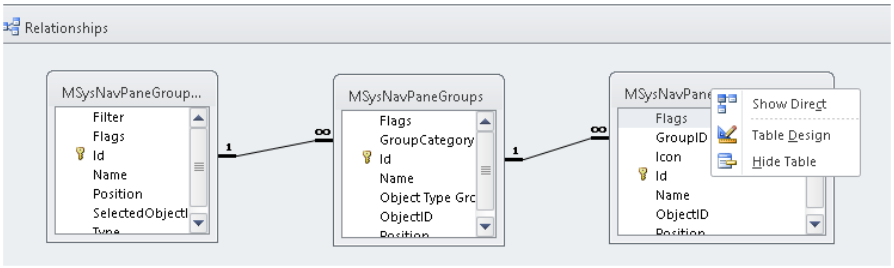
Before explaining that concept, I will explore some of the ways that relationships can be created at table level:

1. Using the Relationships window

Click ***Relationships*** on the ***Database Tools*** ribbon to open the ***Relationships*** window. Depending on your settings, you may find this already contains several ***system tables*** even in a new blank database.



Many developers remove these system tables from the window by hiding these tables.
Right click anywhere on each table and select ***Hide Table***

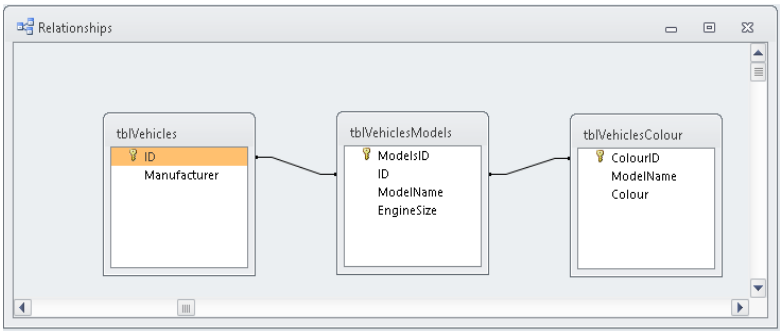


NOTE: this removes the tables from the window but does NOT delete the relationship

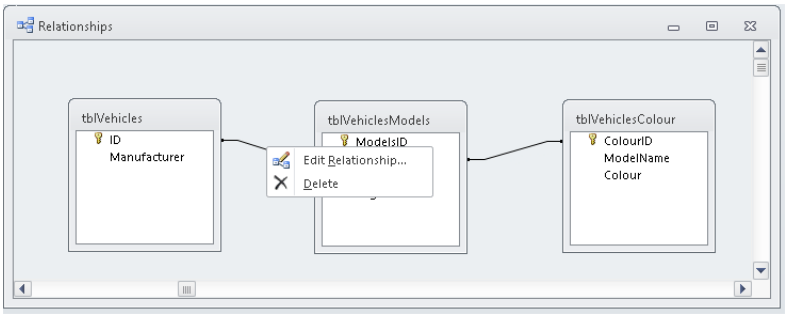
To create new relationships, add two or more linked tables into the window as done in the query designer:

- Right click and select Add Table
- Click the Design tab in the ribbon and select Add Table
- Drag the tables into the window

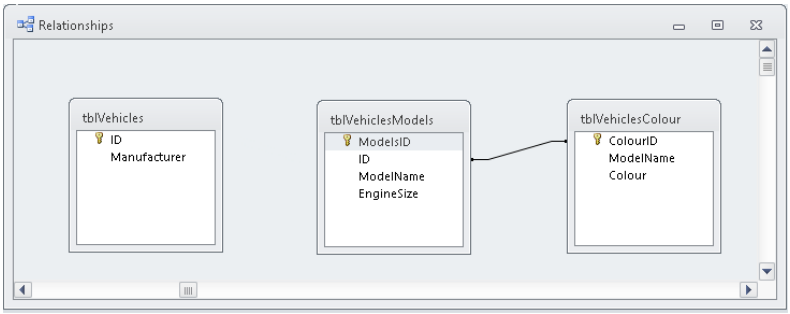
Then join the tables using suitable fields in each table. By default, an inner join is created. For example:



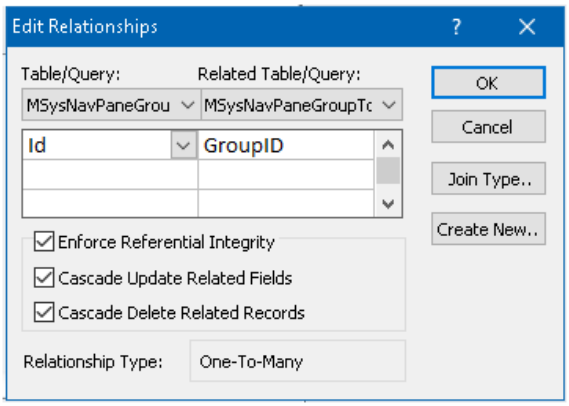
The relationships can be edited or deleted by right clicking on the join line:



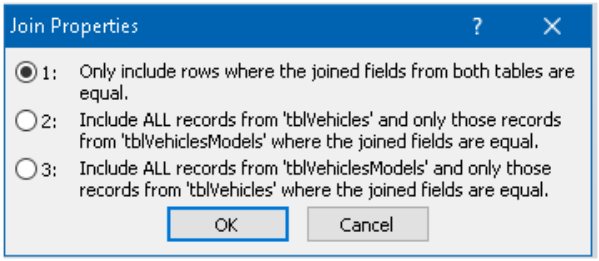
If you click ***Delete***, the relationship is removed.



Click ***Edit Relationship*** to alter the relationship created

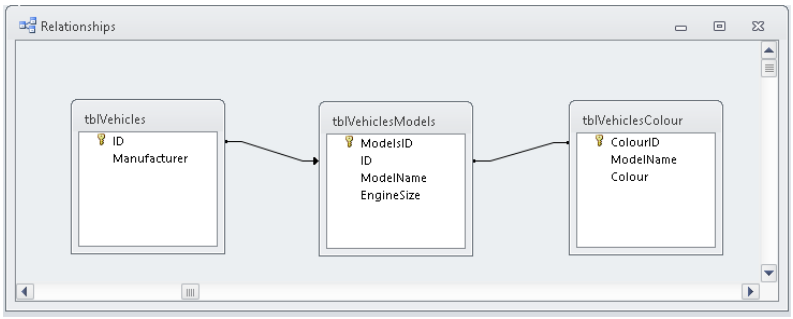


Click ***Join Type*** and a window familiar from the query designer will appear



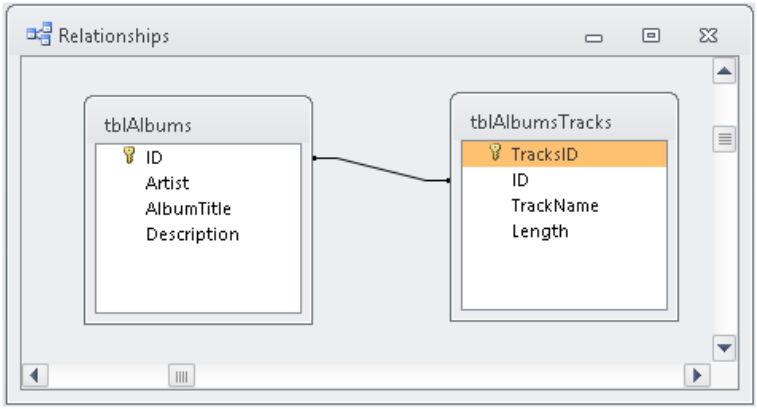
The join types are INNER, LEFT OUTER and RIGHT OUTER. The default is Option 1 (INNER)

If option 2 is chosen, the relationship diagram changes to:

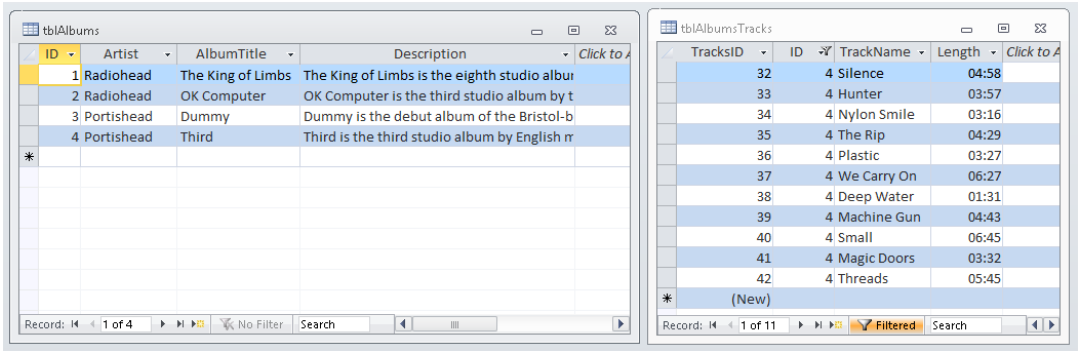


As previously mentioned, the most important reason for using table relationships is to apply ***referential integrity (RI).*** This is used to prevent orphan data remaining in a 'child' table after corresponding data is deleted in the 'parent' table.

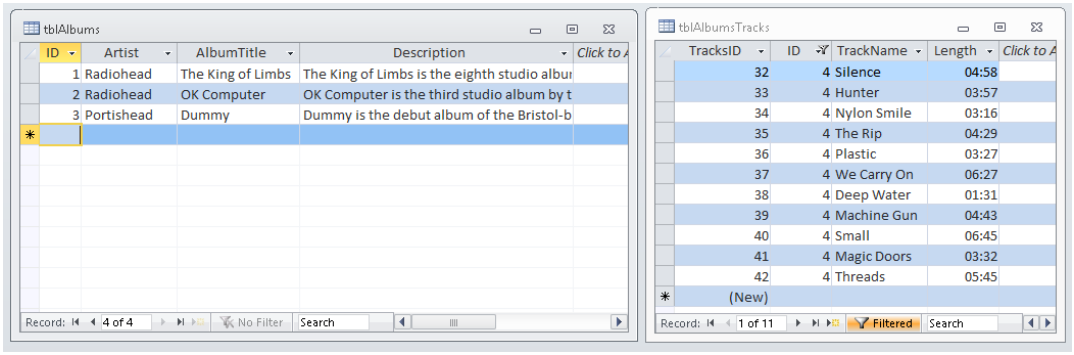
First consider 2 tables joined but without applying referential integrity



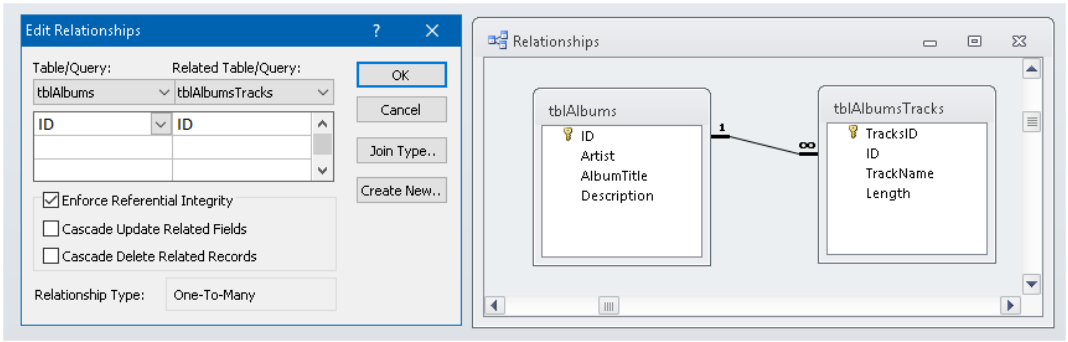
This shows the tracks for the album with ID=4



If that album is deleted in tblAlbums, the corresponding tracks are NOT deleted in tblAlbumTracks. Those records are now orphaned



To add RI, click ***Enforce Referential Integrity*** on the ***Edit Relationships*** window then click ***OK***

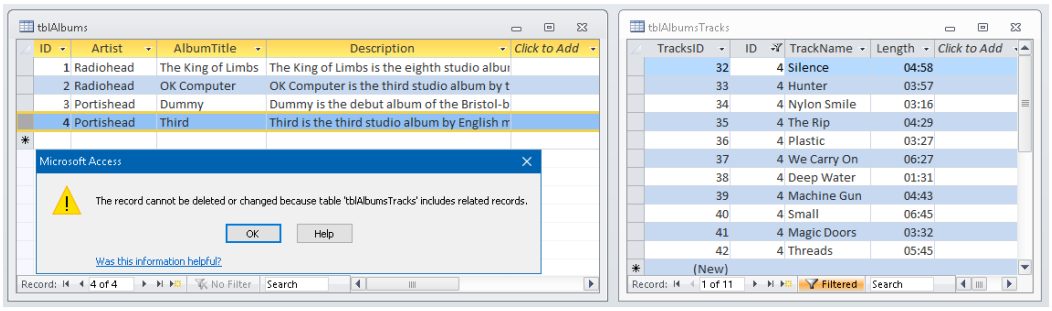


Depending on the fields you have joined, the join line will be marked:

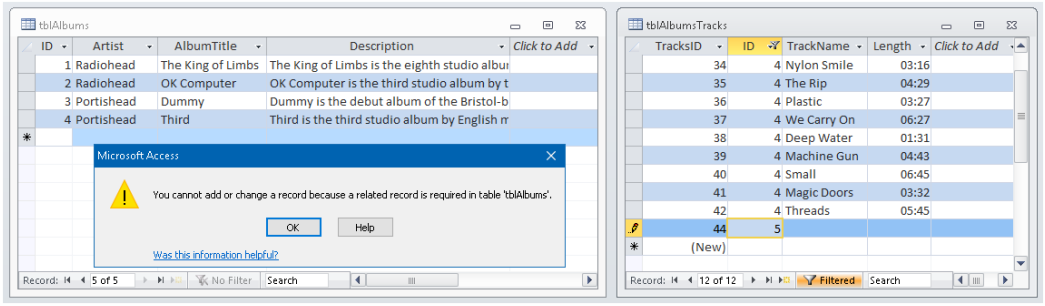
- 1-1 (one to one) where both fields are primary keys
- 1-∞ (1 to many) where one field is not a primary key (so multiple records are possible)

NOTE:
It is best to apply RI before adding data to the tables
You will not be able to enforce RI if one table has records that are missing in the other table

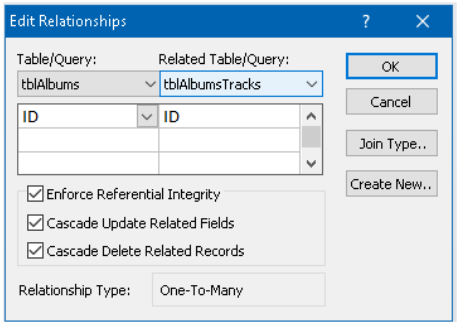
If you try and delete album ID=4 now, Access prevents you doing so as there are related records in tblAlbumTracks



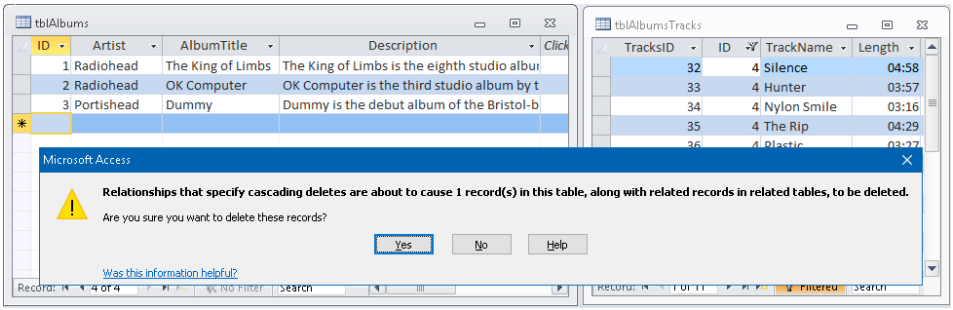
Similarly, it will not allow you to add a record in tblAlbumTracks for a non-existent album 5



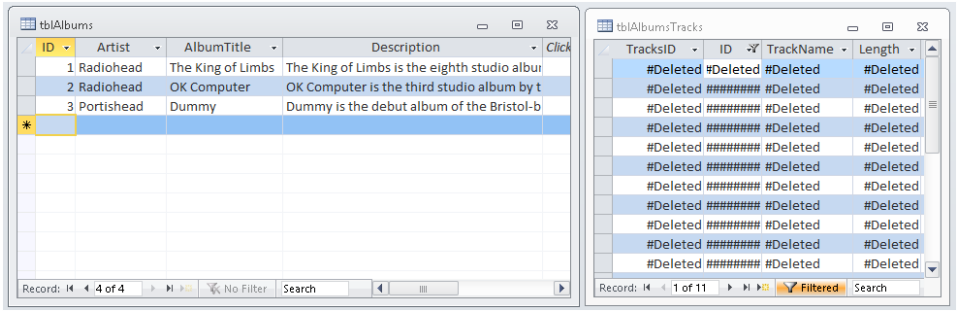
To fix this issue, we need to check the cascade update/delete options.
Doing so, ensures RI is retained when fields are updated and/or records are deleted



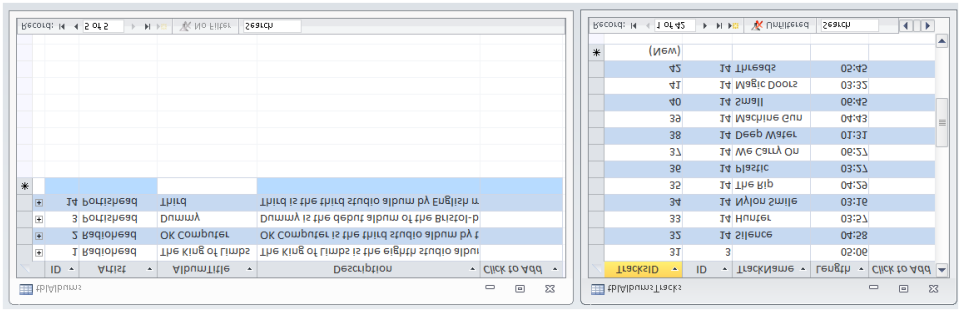
When you try to delete album #4 now, Access warns you of the consequences



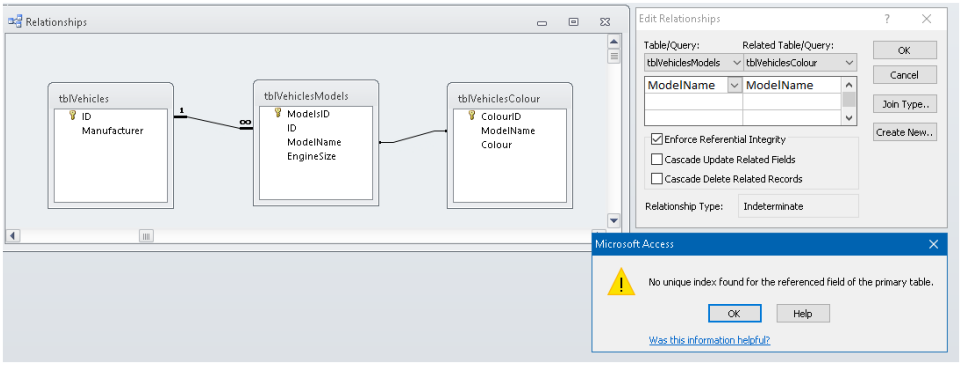
Clicking Yes deletes the corresponding records in both tables



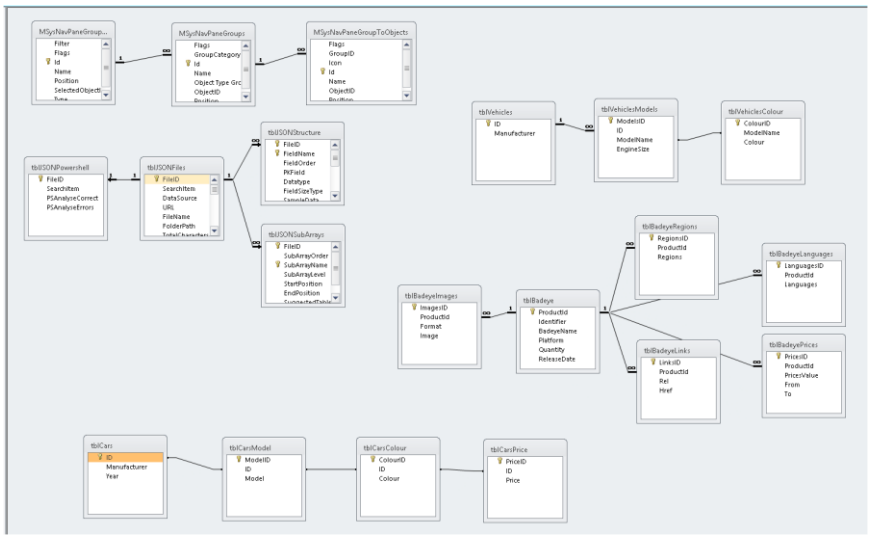
Similarly if album #4 is renumbered as e.g. 14 in tblAlbums, the corresponding field is updated in tblAlbumTracks ensuring that related records remain related



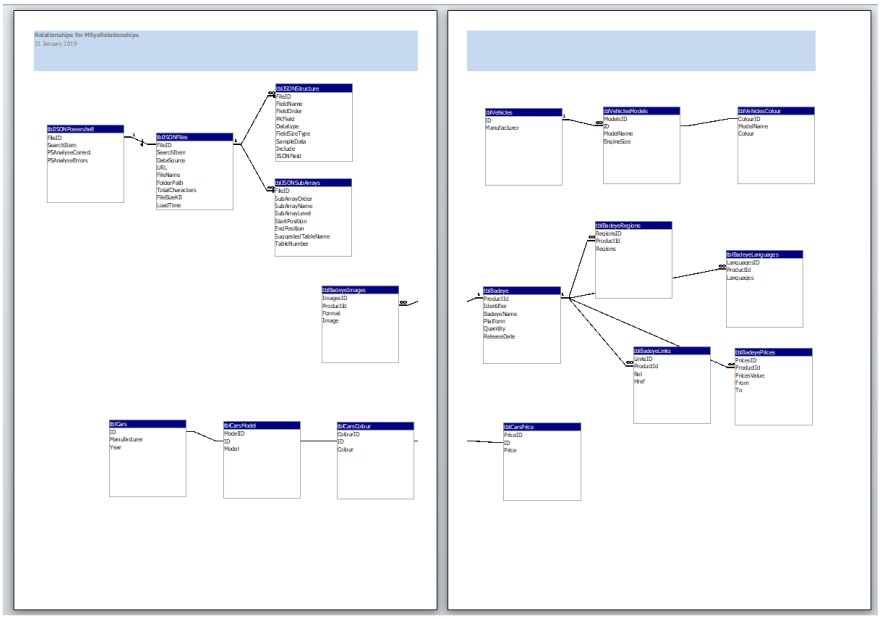
NOTE: It is NOT possible to enforce referential integrity between 2 fields where neither is a primary key field (indeterminate join). This is because there is no unique index



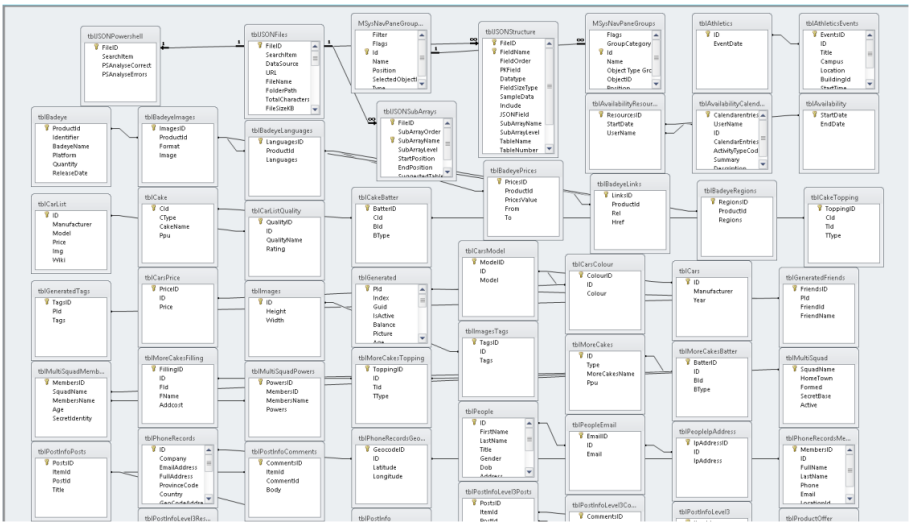
Continue adding relationships as appropriate to other tables in your database



The relationships can also be saved as a report from the Design ribbon when the relationships window is displayed. However, the report layout isn't very good and is difficult to modify



For large databases containing many tables, the relationships window can become very crowded

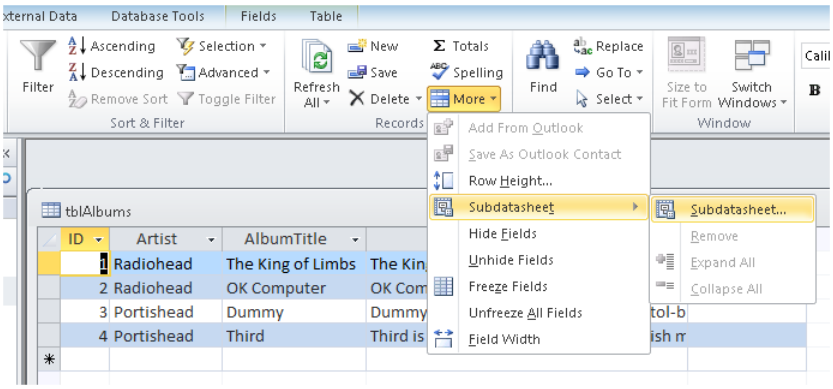


You can move/shrink items to help improve the layout to some extent.
By this stage, you may wish to hide some of the tables without deleting the relationships between them

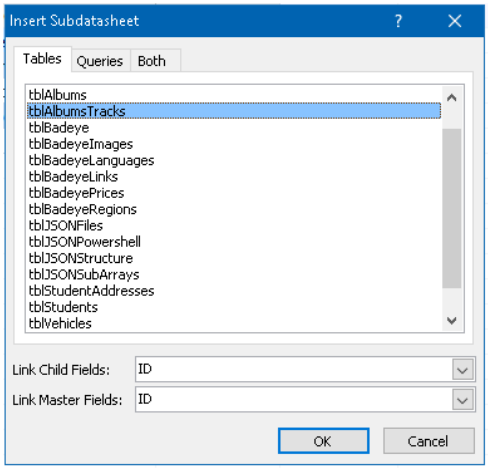
NOTE:
For linked tables, the relationships need to be created in the linked backend database
You can display a backend table relationship in the frontend relationships window.
However, doing this will not override any relationship already created in the backend

2. *Using subdatasheets*

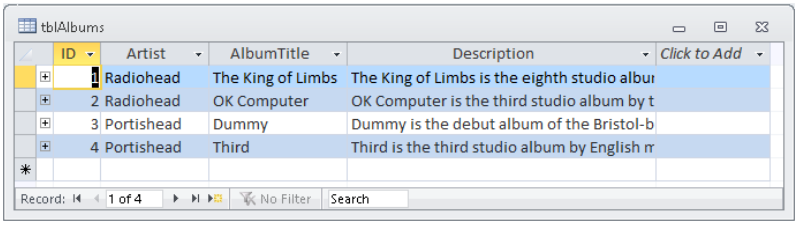
Another way of adding relationships is to use subdatasheets on the Home tab



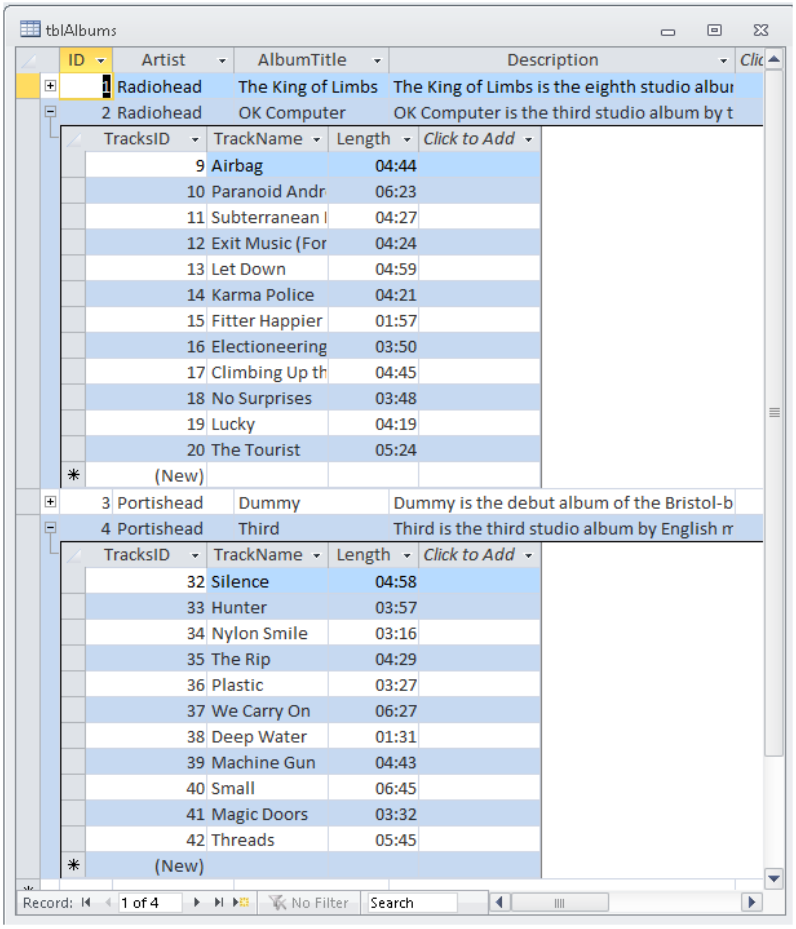
Select the table to be linked and the Master/Child fields then click OK



The table will now show a + sign indicating it has a linked subdatasheet

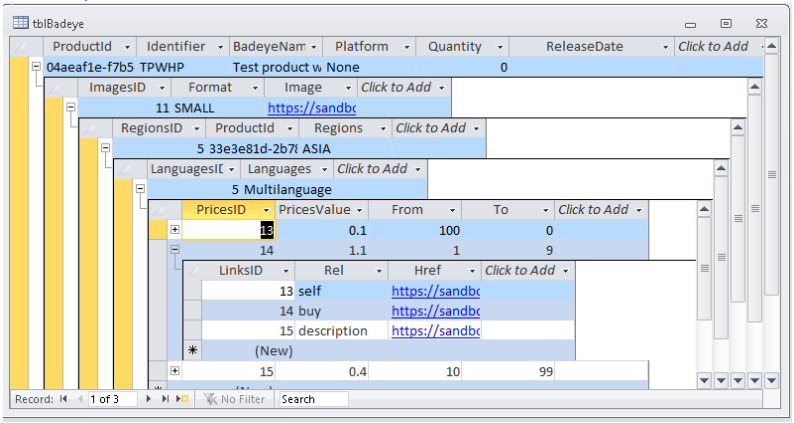


Click the + to show the subdatasheet for one or more records

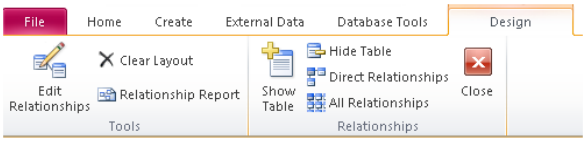


Click the – to close the subdatasheet again

If you have several tables that can be joined, this can be repeated to create cascading subdatasheets for all tables you have linked



However, the use of subdatasheets can be very confusing to end users.
Subdatasheets also slow form loading as each subdatasheet has to be loaded when the form opens
For those reasons, many developers do not use subdatasheets.
Relationships created using subdatasheets do not automatically appear in the relationships window
To display relationships created using subdatasheets, click All Relationships on the Design ribbon



Similarly, relationships created from the relationships window do not automatically create subdatasheets

3. Managing relationships using code

It is also possible to add (or delete) relationships using VBA.
For example, use this code to create a relationship between the 2 tables used above

```
Function CreateRelationship()

On Error GoTo Err_Handler
   Dim db As DAO.Database
   Dim rel As DAO.Relation
   Dim fld As DAO.Field

   'Initialize
   Set db = CurrentDb()

   'Create a new relationship.
   Set rel = db.CreateRelation("tblAlbumstblAlbumsTracks")

   'Define its properties.
   With rel
      'Specify the primary table.
      .Table = "tblAlbums"
      'Specify the related table.
      .ForeignTable = "tblAlbumsTracks"
      'Specify attributes for cascading updates and deletes.
      .Attributes = dbRelationUpdateCascade + dbRelationDeleteCascade

      'Add the fields to the relationship.
      'Field name in primary table.
      Set fld = .CreateField("ID")
      'Field name in related table.
      fld.ForeignName = "ID"
      'Append the field.
      .Fields.Append fld

      'Repeat for other fields if a multi-field relation.
   End With

   'Save the newly defined relation to the Relations collection.
   db.Relations.Append rel
    'Debug.Print "Relationship created."

   'Clean up
   Set fld = Nothing
   Set rel = Nothing
   Set db = Nothing

Exit_Handler:
   Exit Function

Err_Handler:
   'error 3012 if relationship already exists
   MsgBox "Error " & Err.Number & " in CreateRelationship procedure : " & _
      Err.description, vbCritical, "Program error"
   Resume Exit_Handler

End Function
```

This code deletes the relationship if it exists

```
Function DeleteRelationship()

On Error GoTo Err_Handler

   DBEngine(0)(0).Relations.Delete "tblAlbumstblAlbumsTracks"

Exit_Handler:
   Exit Function

Err_Handler:
   'err 3265 if relationship doesn't exist
   MsgBox "Error " & Err.Number & " in DeleteRelationship procedure : " & _
      Err.description, vbCritical, "Program error"
   Resume Exit_Handler
End Function
```

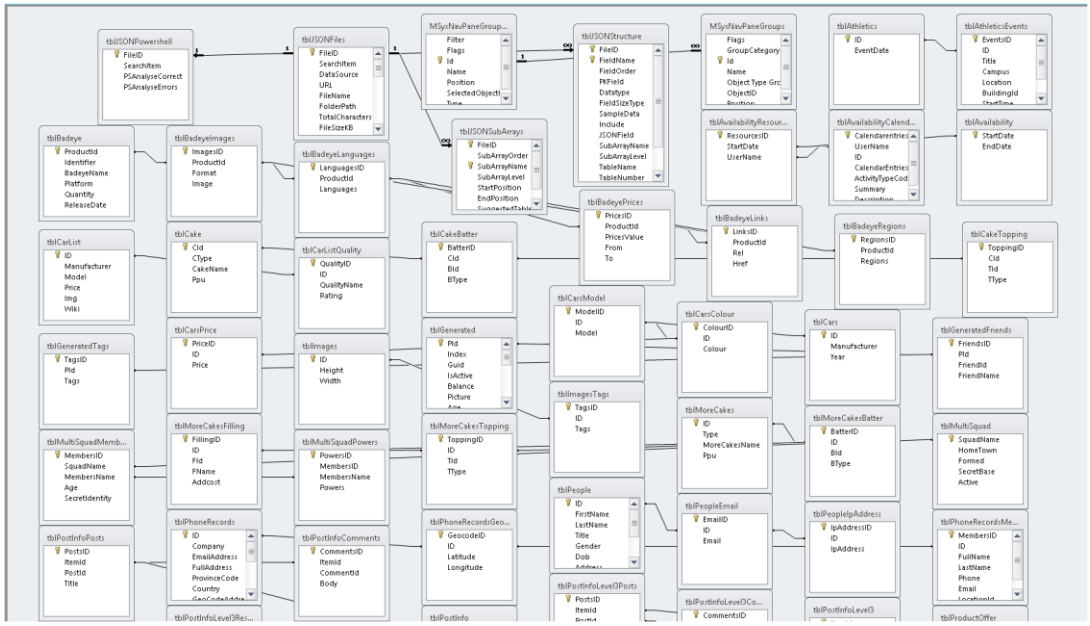
For more details and example code, see http://allenbrowne.com/func-DAO.html#CreateRelationDAO

The second part of this article explores how the relationship details are stored by Access using a hidden system table ***MSysRelationships***

Part 2 – Analysing Relationships

The first part of this article explored how relationships are used with particular reference to referential integrity.

The article also discussed how the relationships window can get overcrowded resulting in the need to hide tables:



However, if the tables are hidden, how can we still keep a check on the relationships in use.
The answer is to query the hidden system table MSysRelationships where the information is stored

By default, this does not appear in the navigation pane. To make the table visible temporarily, tick Show Hidden Objects and Show System Objects in Navigation Options

The table contains 8 fields – two of these (**ccolumn** and **icolumn**) can be ignored

MSysRelationships

ccolumn	grbit	icolumn	szColumn	szObject	szReferencedColumn	szReferencedObject	szRelationship
1	16781568	0	FileID	tblJSONSubArrays	FileID	tblJSONFiles	{97F0F179-DC30-4DE3-85CB-ADA081714FA1}
1	16781568	0	FileID	tblJSONStructure	FileID	tblJSONFiles	{E5E86A18-7F8C-4DDD-AAA2-3B8A5B49DC10}
1	4352	0	GroupCategon	MSysNavPaneGroups	Id	MSysNavPaneGroupCategories	MSysNavPaneGroupCategoriesMSysNavPaneGroups
1	4352	0	GroupID	MSysNavPaneGroupToObjects	Id	MSysNavPaneGroups	MSysNavPaneGroupsMSysNavPaneGroupToObjects
1	4352	0	ProductId	tblBadeyeImages	ProductId	tblBadeye	tblBadeyetblBadeyeImages
1	4352	0	ProductId	tblBadeyeLanguages	ProductId	tblBadeye	tblBadeyetblBadeyeLanguages
1	4352	0	ProductId	tblBadeyeLinks	ProductId	tblBadeye	tblBadeyetblBadeyeLinks
1	4352	0	ProductId	tblBadeyePrices	ProductId	tblBadeye	tblBadeyetblBadeyePrices
1	4352	0	ProductId	tblBadeyeRegions	ProductId	tblBadeye	tblBadeyetblBadeyeRegions
1	16781569	0	FileID	tblJSONPowershell	FileID	tblJSONFiles	tblJSONFilestblJSONPowershell
1	0	0	ID	tblVehiclesModels	ID	tblVehicles	tblVehiclestblVehiclesModels
1	2	0	ModelName	tblVehiclesColour	ModelName	tblVehiclesModels	tblVehiclesModelstblVehiclesColour

Record: 1 of 12 | No Filter | Search

The **MSysRelationships** table is READ ONLY so no damage can be done.
However, it is better to create a query then hide the system tables again.
The MSys tables themselves are excluded from this query and aliases used to clarify the purpose of each field

```
SELECT MSysRelationships.szRelationship AS RelationshipName, MSysRelationships.szObject AS TableName,
MSysRelationships.szColumn AS FieldName, MSysRelationships.szReferencedObject AS ParentTableName,
MSysRelationships.szReferencedColumn AS ParentFieldName, MSysRelationships.grbit AS RelValue
FROM MSysRelationships
WHERE (((MSysRelationships.szObject) Not Like 'MSys*'));
```

For example:

qryRelationships

RelationshipName	RelValue	ParentTableName	ParentFieldName	TableName	FieldName
tblBadeyetblBadeyeImages	4352	tblBadeye	ProductId	tblBadeyeImages	ProductId
tblBadeyetblBadeyeLanguages	4352	tblBadeye	ProductId	tblBadeyeLanguages	ProductId
tblBadeyetblBadeyeLinks	4352	tblBadeye	ProductId	tblBadeyeLinks	ProductId
tblBadeyetblBadeyePrices	4352	tblBadeye	ProductId	tblBadeyePrices	ProductId
tblBadeyetblBadeyeRegions	4352	tblBadeye	ProductId	tblBadeyeRegions	ProductId
tblJSONFilestblJSONPowershell	16781569	tblJSONFiles	FileID	tblJSONPowershell	FileID
{E5E86A18-7F8C-4DDD-AAA2-3B8A5B49DC10}	16781568	tblJSONFiles	FileID	tblJSONStructure	FileID
{97F0F179-DC30-4DE3-85CB-ADA081714FA1}	16781568	tblJSONFiles	FileID	tblJSONSubArrays	FileID
tblVehiclesModelstblVehiclesColour	2	tblVehiclesModels	ModelName	tblVehiclesColour	ModelName
tblVehiclestblVehiclesModels	0	tblVehicles	ID	tblVehiclesModels	ID

Record: 10 of 10 | No Filter | Search

The query shows the relationship name together with the table and field names being joined.
However, the important field here is a long integer field '***grbit***' which has been given the alias ***RelValue***
The values indicate the type of join used and whether referential integrity is being enforced

The 'base value' = 0 is for an inner join with referential integrity using PK field in one table and a foreign key in the other

Relationship/Join Type	grbit
1 to many (PK to FK)	0
1:1 join (PK to PK)	1
Referential Integrity not enforced	2
Cascade Update	256
Cascade Delete	4096
Left Outer	16777216
Right Outer	33554432

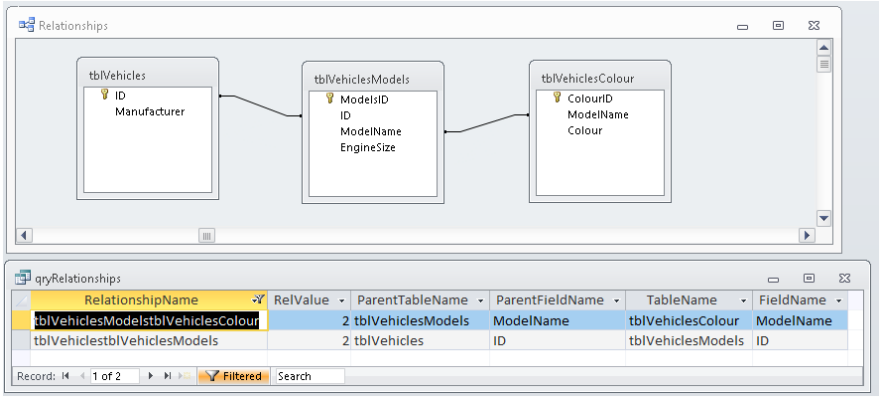
The ***grbit*** values are cumulative. Some examples include:

Join Type	Join Value	PK field Join	RI	Cascade Update	Cascade Delete	Relationship Calculation	Rel Value (grbit)
Inner	Indeterminate	Neither	No	N/A	N/A	2	2
Inner	1-many	One	No	N/A	N/A	2	2
Inner	1-many	One	Yes	No	No	0	0
Inner	1-many	One	Yes	Yes	No	0 + 256	256
Inner	1-many	One	Yes	No	Yes	0 + 4032	4096
Inner	1-many	One	Yes	Yes	Yes	0 + 256 + 4096	4352
Inner	1-1	Both	No	N/A	N/A	1 +2	3
Inner	1-1	Both	Yes	No	No	1	1
Inner	1-1	Both	Yes	Yes	No	1+256	257
Inner	1-1	Both	Yes	No	Yes	1+4096	4097
Inner	1-1	Both	Yes	Yes	Yes	1+256+4096	4353
Left	Indeterminate	Neither	No	N/A	N/A	16777216 + 2	16777218
Left	1-many	One	No	N/A	N/A	16777216 + 2	16777218
Left	1-many	One	Yes	No	No	16777216 + 0	16777216
Left	1-many	One	Yes	Yes	Yes	16777216 +0 +256 +4096	16781568
Left	1-1	Both	Yes	No	No	16777216 + 1	16777217
Left	1-1	Both	Yes	Yes	No	16777216 + 1 + 256	16777453
Left	1-1	Both	Yes	No	Yes	16777216 + 1 + 4096	16781313
Left	1-1	Both	Yes	Yes	Yes	16777216 + 1 + 256 + 4096	16781569
Right	Indeterminate	Neither	No	N/A	N/A	3355432 + 2	33554434
Right	1-many	One	No	N/A	N/A	3355432 + 2	33554434
Right	1-many	One	Yes	No	No	3355432 + 0	33554432
Right	1-many	One	Yes	Yes	Yes	33554432 + 256 + 4096	33558784
Right	1-1	Both	Yes	No	No	33554432 + 1	33554433
Right	1-1	Both	Yes	Yes	No	33554432 + 1 + 256	33554689
Right	1-1	Both	Yes	No	Yes	33554432 + 1 + 4096	33558529
Right	1-1	Both	Yes	Yes	Yes	33554432 + 1 + 256 + 4096	33559785

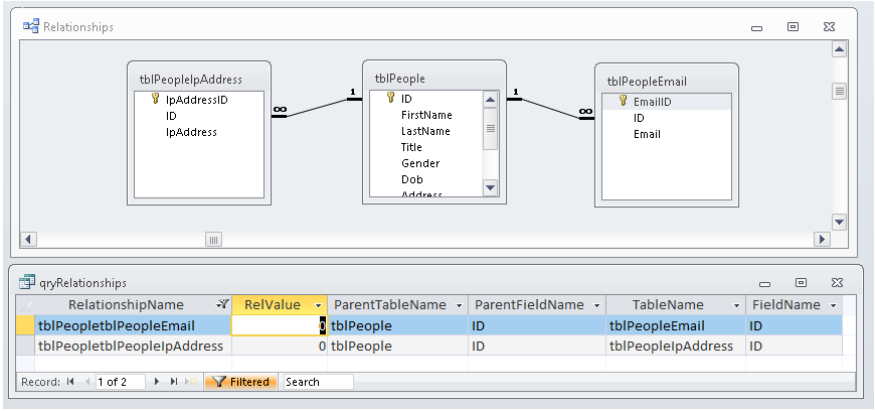
NOTE: A SELF join linking PK field to another field will be 1 to many
Hence an inner self join = 1 (using RI) or 3 (without RI)

Some examples:

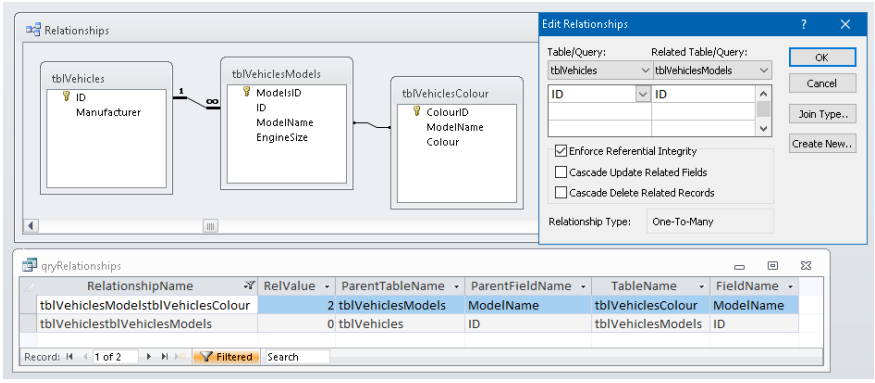
a) 2 inner joins – no RI



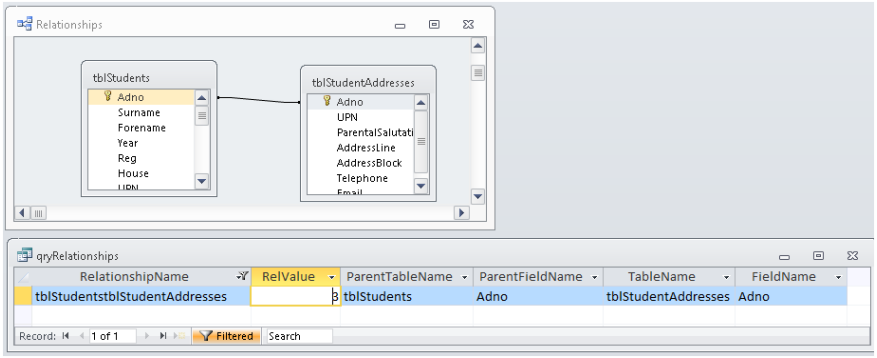
b) 2 inner joins with RI (1-many) but no cascade update / cascade delete



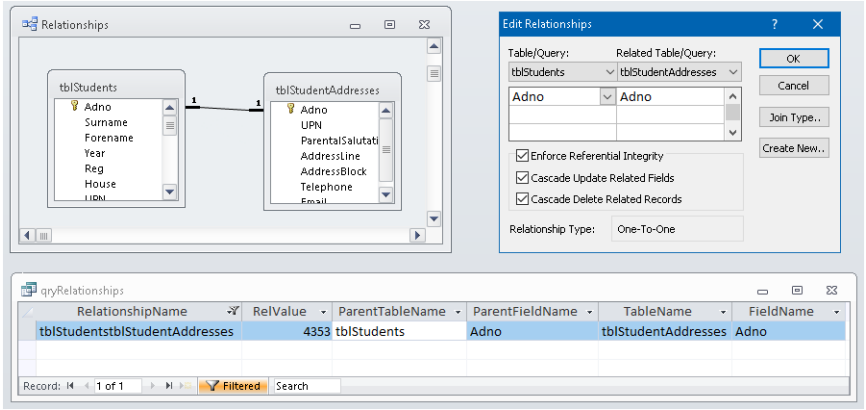
c) 2 inner joins – one with 1-many RI but not cascade update/cascade delete ; other Indeterminate



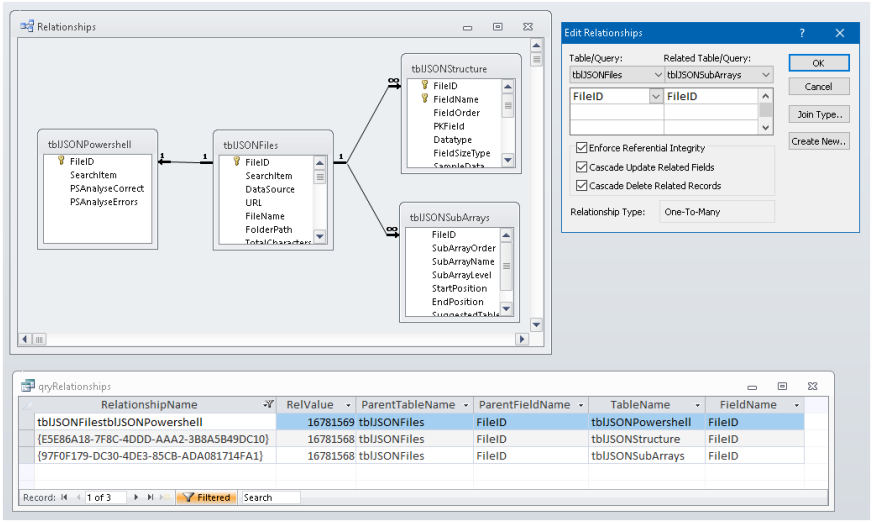
d) Inner 1:1 join on PK field– no RI



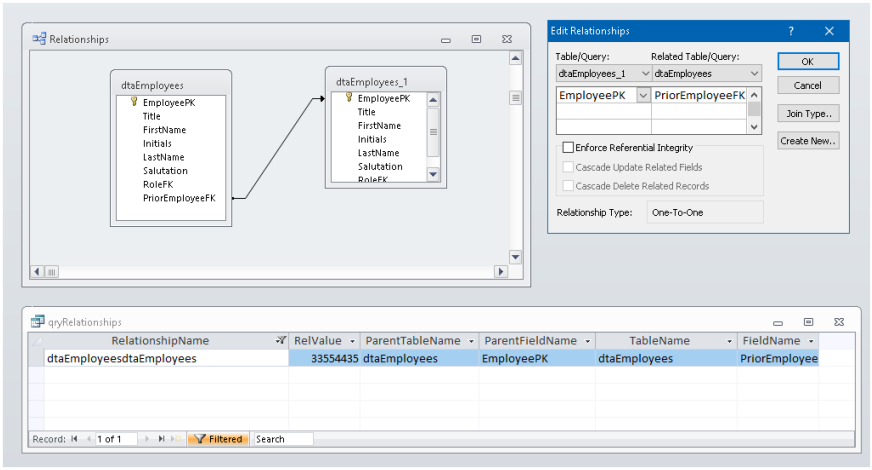
e) Inner 1:1 join on PK field – with RI and Cascade Update / Cascade Delete



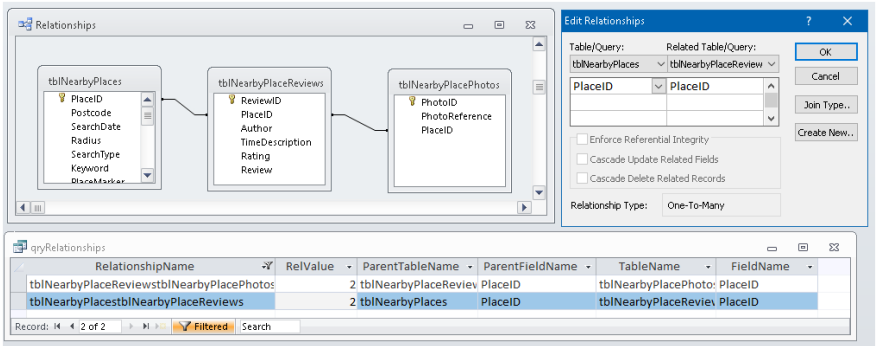
f) 3 left joins (one 1-1; two 1-many) ; all with RI and cascade update/delete



g) Outer self join without RI



h) 2 inner joins on linked tables – RI cannot be enforced in the front end db



If you are interested, you can use the supplied query **qryRelationships** in your own databases to explore other relationship types not covered above

I hope the above article has added to your understanding of **relationships** including the purpose of **referential integrity** as well as explaining how Access stores this information in the **MSysRelationships** system table

Colin Riddington ***Mendip Data Systems*** ***Updated 29 Sept 2019***