

Section 4

Relationships

By the end of this Section you should be able to:

Apply Primary Keys

Apply and Modify Different Types of Relationship

Query Related Tables

Understand Joins

Apply Referential Integrity

Update and Delete Related Records

Work with Subdatasheets

To gain an understanding of the above features, work through the **Driving Lessons** in this **Section**.

For each **Driving Lesson**, read the **Park and Read** instructions, without touching the keyboard, then work through the numbered steps of the **Manoeuvres** on the computer. Complete the **Revision Exercise(s)** at the end of the section to test your knowledge.

Driving Lesson 23 - Applying a Primary Key

Park and Read

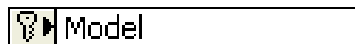
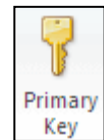
When creating relational databases which use more than one table, it is important to be able to uniquely identify individual records in a table if they are to be referenced from another table. Records are usually identified by specifying a field in the table which contains unique data for each record, e.g. a serial number or identification number. This is then defined as the **Primary Key**.

Use of a **Primary Key** prevents duplication of records in a table and also allows sorting and querying to be performed more efficiently. It also enables the linking of tables.



Manoeuvres

1. Open the **Custom Computers** database and then the **Computers** table in **Design View**. Make sure the **Design** tab is selected.
2. Click in the **Model** field. Click on the **Primary Key** button in the **Tools** group on the **Design** tab.



The **Primary Key** is applied to the **Model** field. Notice in the field properties that the **Indexed** property is set to **Yes (No Duplicates)**. This means that there can be no duplicate **Model** in any two records within this table.

3. This is obviously a bad choice for the **Primary Key** field as it is likely that there will be more than one record in the table for any particular model of computer. Click in the **Serial Number** field and click on the **Primary Key** button again.
4. **Serial Number** is now designated as the **Primary Key** field. This is an ideal choice, as serial number is a unique identifier for any particular computer.



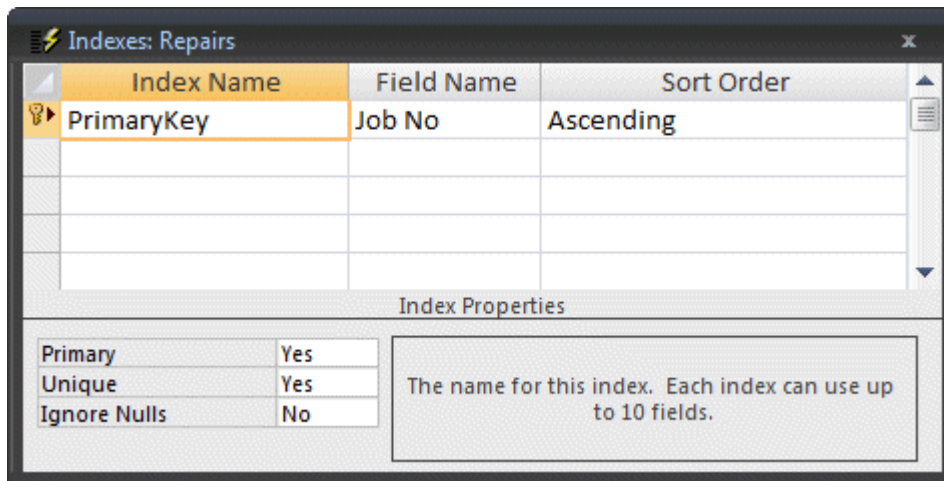
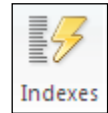
When **Serial Number** was selected as the **Primary Key**, the indicator was automatically removed from the **Model** field.

5. **Save** and close the table.
6. Open the **Repairs** table in **Design View**. A **Primary Key** is already applied to the **Job No** field.



Driving Lesson 23 - Continued

7. Notice that the **Repairs** table also contains the computer serial number field, but on this table it is not necessarily unique. There may be several repair jobs for the same computer.
8. A **Primary Key** is just a special index for a table. Click the **Indexes** button on the **Design** tab, to view the **Primary Key** information.



The **Index Properties** show that a **Primary Key** field is unique and that **Ignore Nulls** is set to **No**. This means that as well as being unique, a primary key field cannot be left blank in a record.

9. Close the **Indexes** window and the **Repairs** table.
10. Leave the **Custom Computers** database open.

Driving Lesson 24 - Applying Relationships

Park and Read

Once tables have been designed and primary keys applied, a **Relationship** may be applied between two or more tables to link them together. Once two or more tables are linked by a relationship, the data from all of the tables may be used to create a single query, form or report.

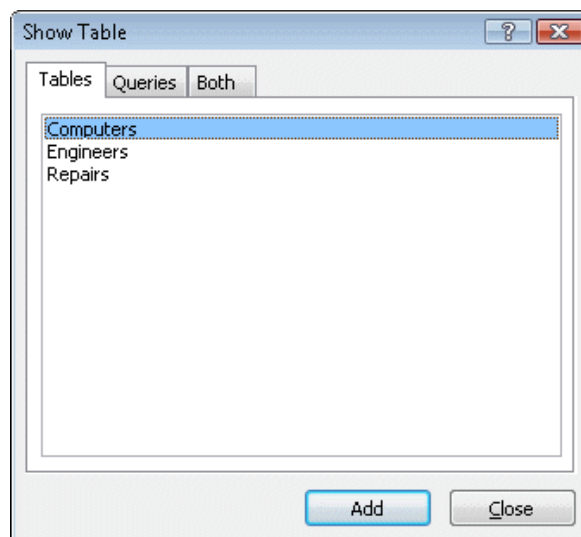
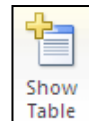
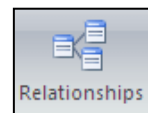
Relationships are usually applied between tables which contain a common field. Usually, the related field in the first table is the primary key field and is known as the **Primary Table**. In this way, a record in one table can link to further information on another table. The related field in the second table is known as the **Foreign Key**.

Applying relationships allows many smaller tables to be linked together to form the complete database, improving its overall efficiency. In the Driving Lesson below then, if the **Computer** and **Repairs** tables are linked using the common field serial number, there is no need to have all computer details on every repair record. A query on the **Repairs** table will use **Serial Number** to access all related data on the **Computers** table automatically.

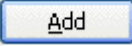
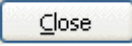
This lesson describes the most common type of relationship, **One to Many**. Other types of relationships are introduced over the next few Driving Lessons.

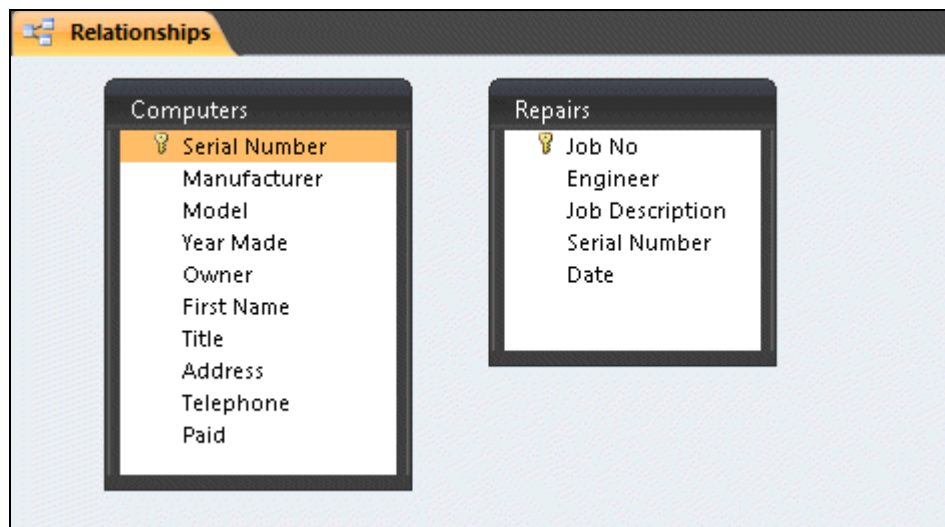
Manoeuvres

1. Display the **Database Tools** tab and click the **Relationships** button. A blank **Relationship** window will be displayed.
2. If the **Show Table** dialog box is not shown, click the **Show Table** button, on the **Design** tab. This displays the available tables in this database.

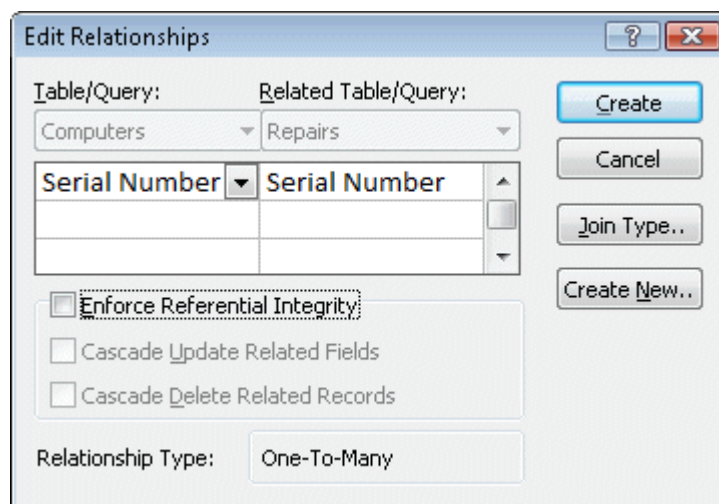


Driving Lesson 24 - Continued

3. With the **Computers** table highlighted, click  to place the table in the **Relationships** window.
4. Click on the **Repairs** table and again add it to the window then click  to remove the **Show Table** dialog box.
5. Resize the table boxes to see all of their fields. Notice the **Primary Keys** for each table indicated with a key symbol.



6. Highlight the **Serial Number** field in the **Computers** table.
7. Drag the **Serial Number** field, from the **Computers** table, over the **Serial Number** field (the foreign key) in the **Repairs** table. Release the mouse when in position.

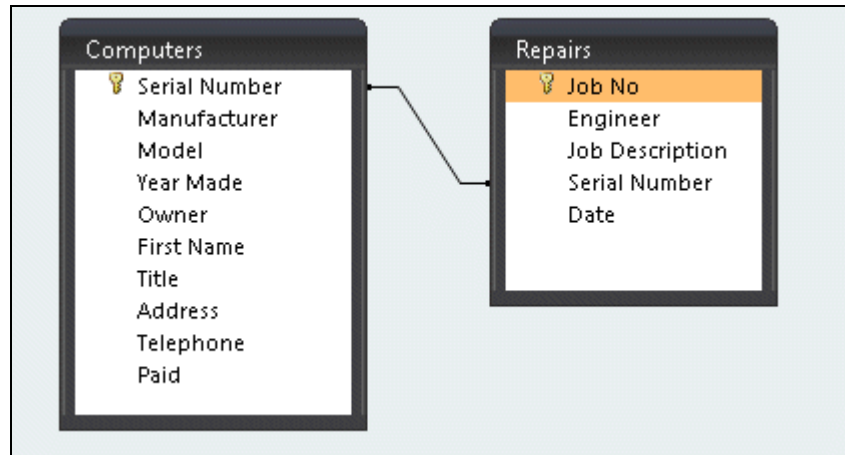


8. In the **Edit Relationships** dialog box, note the relationship type is **One-To-Many**. This is the most common type of relationship.



Driving Lesson 24 - Continued

9. Click to create the relationship.
10. Notice how the relationship between the tables is now symbolised by a line, linking the same field, **Serial Number**.



The relationship is **One-To-Many**. This means that **one** record from **Computers** can have **many** related records in **Repairs**, i.e. one computer may have had several repairs. The fact that one of the fields in the link is a primary key and the other is not, defines the relationship as **One-To-Many**.

11. To delete the relationship, right click on the connecting line and select **Delete**. Select **Yes** to confirm the deletion.



Any relationship can be modified by right clicking on the connecting line and selecting **Edit Relationship**.

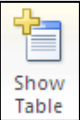
12. Now click and drag from **Owner** in the **Computers** table to **Engineer** in the **Repairs** table.
13. This is not a valid relationship; the relationship type is shown as **Indeterminate**. Click **Cancel**.
14. Reapply the original relationship between **Serial Number** in both tables and click **Create**.
15. Close the **Relationships** window and select **Yes** when prompted to save the changes to the relationship.
16. Leave the database open for the next Driving Lesson.

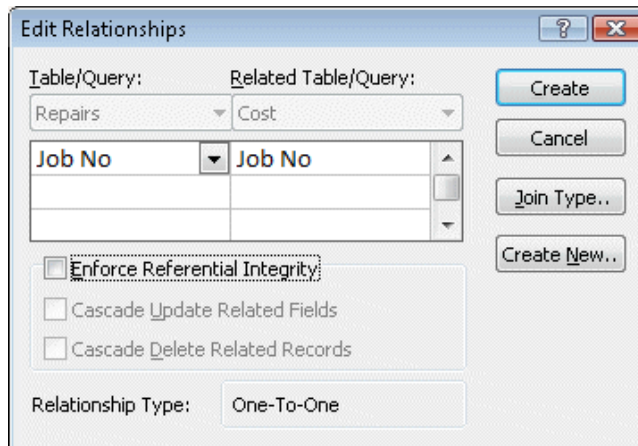
Driving Lesson 25 - One-to-One Relationships

Park and Read

A **One-to-One** relationship is used where one record in one table is linked to only one record in another. This can be used to split a large table with many fields, or if part of a table needs to be removed for security reasons, or if the second table contains optional data which is not always required.

Manoeuvres

1. To create a **One-to-One** relationship in the **Custom Computers** database, create a table in **Design** view with two fields, **Job No** and **Charge**.
2. The **Job No** field will have a data type of **Number** and the **Charge** field will be **Currency**.
3. Make the **Job No** field the **Primary Key** and save the table as **Cost**. Close it without adding any data.
4. Open the **Relationships** window showing the existing relationship and use the **Show Table** button to display the dialog box. 
5. Select **Cost** then click **Add** to add it to the **Relationships** window. Close the **Show Table** box.
6. Reposition the **Cost** table if necessary, then make the link between **Job No** in the **Repairs** table and **Job No** in the **Cost** table. Because each of these fields is a primary key, this time in the **Edit Relationships** box, the **Relationship Type** is **One-To-One**.



7. Click **Create**.



As with any relationship, this can be deleted or modified by right clicking on the connecting line and selecting **Delete** or **Edit Relationship**.





Driving Lesson 26 - Many-to-Many Relationships

Park and Read



A **Many-to-Many** relationship is used when a record in the first table can have many matching records in the second, and vice versa. For example, a single product may have many orders and a single order may be for many products. A single **Many-to-Many** relationship cannot exist. An intermediate **junction table** must be created with **One to Many** links to the two original tables. It must contain two fields: the foreign keys from both tables.

Manoeuvres

1. Open the **CiA** database. To create a **Many-to-Many** relationship between the **Orders** table and the **Products** table it is first necessary to create an intermediate, junction table.
2. Create a new table in **Design View** containing the fields **Order Ref (Number)** and **Product Ref (Text)**.
3. The primary key for this new table will be defined as the combination of both fields (neither individual field will be unique on the table). Click in the area at the left of **Order Ref** to highlight the whole row.
4. Hold down <Ctrl> and click at the left of **Product Ref**. Both rows will be highlighted.
5. Click the **Primary Key** button.

	Field Name	Data Type
	Order Ref	Number
	Product Ref	Text

6. Save the table as **Junction** and close it.
7. This table would now have to be filled with information about which products were on which orders. Fortunately such a table already exists in the database. Open the **Order Details** table in **Datasheet View**, showing the information on each order line.
8. Switch to **Design View**.

	Field Name	Data Type	
	Order Ref	Number	Order reference
	Product Ref	Text	Product ordered
	Line Number	Number	Order Line
	Amount	Number	Quantity ordered

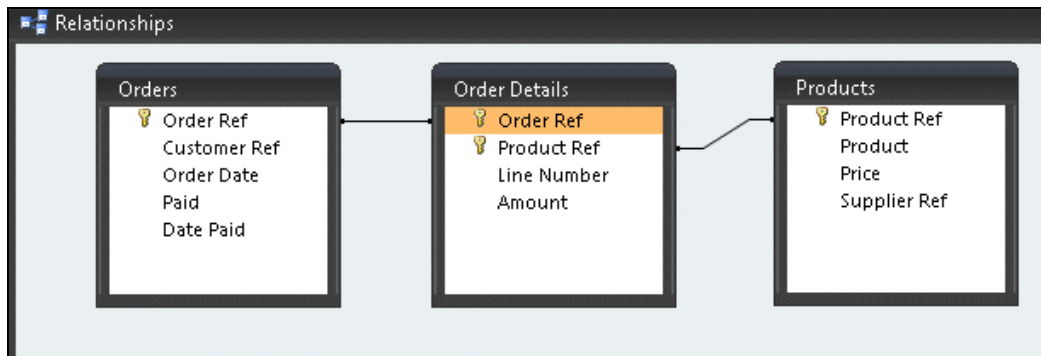


Driving Lesson 26 - Continued



In a working database, this table would be maintained as part of the **Order Entry** process.

9. Close the table and view the **Relationships** window. Place the **Orders**, **Order Details** and **Products** tables on to it.
10. Create a **One-to-Many** relationship between the **Orders** table and the **Order Details** table using the **Order Ref** field.
11. Create a **One-to-Many** relationship between the **Products** table and the **Order Details** table using the **Product Ref** field. The overall effect now is that there is a **Many-to-Many** relationship between the **Orders** and **Products** tables.



As with any relationship, these can be deleted or modified by right clicking on the connecting line and selecting **Delete** or **Edit Relationship**.

12. Close the **Relationships** window, saving when prompted.
13. Create a query based on the three tables shown above.
14. Add **Customer Ref** and **Product** to the grid and specify an **Ascending Sort** on **Customer Ref**.
15. Run the query to see which products have been bought by each customer.
16. Switch to **Design View**, remove the **Customer** sort and specify an **Ascending Sort** on the **Product**.
17. Run the query again to see which customers have bought each product.
18. Close the query without saving and close the database.

Driving Lesson 27 - Applying Joins

Park and Read

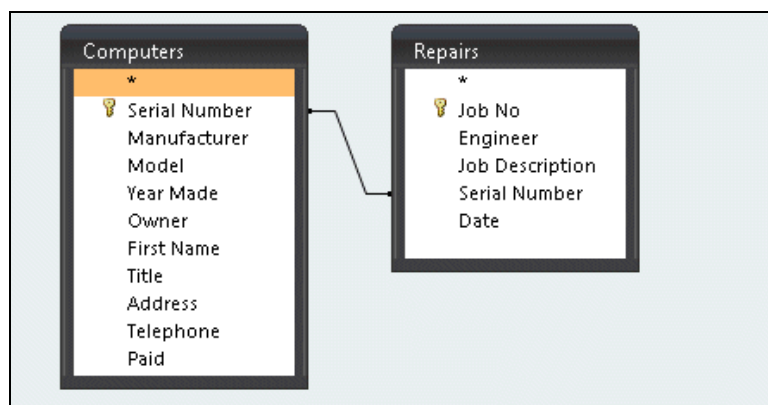
Joins describe the links between tables. They affect the way queries select records when related tables are involved.

The default join type, which is applied by Access automatically, is an **inner join**. With this type of join, a query will only display records where there is a corresponding entry in both tables. For example, a query on linked **Computers** and **Repairs** tables would only show computers that had a repair. Any computers without a repair job, or jobs without corresponding computer records, would not be displayed. This is an important point to remember. Sometimes a linked tables query will not display all the records you expect it to because of this.

Using the above example, if you wanted to display all computers in a query, whether or not they had a match in the **Repairs** table, you would redefine the link as an **outer join** based on the **Computers** table. An alternative **outer join** based on the **Repairs** table could also be defined, which would show all repair records, even when there was no associated computer record. A **subtract join** is the opposite of an outer join; it includes only those records in one table that don't match any record in the other table.

Manoeuvres

1. Open the **Custom Computers** database and open the **Computers** table.
2. Create the following record: **B12345A, Bantacom, Pentio, 2007, Smallfoot, Andrew, Mr, 234 Cedar Drive**. Leave all other fields blank.
3. Close the table. The computer is registered but has had no repairs yet.
4. Create a new query in **Design View**. Add the **Computers** table and then the **Repairs** table to the grid. Because there is a relationship defined for these tables the link will be displayed.



Driving Lesson 27 - Continued

5. Double click the relationship line to display the **Join Properties** dialog box.



6. Notice how join type **1** is selected by default. This is an **inner join**. Click **OK** without making any changes.
7. From **Computers**, place **Serial Number** and **Model** on the grid.
8. From **Repairs**, place **Job No**, **Engineer** and **Date** on the grid.
9. Run the query. Even though the new **Bantacom Pentio** record is in the **Computers** table, it is not shown in the results because only records with repairs are displayed.
10. Switch back to **Design View** and double click the relationship line. To define the link as a **left outer join** based on the **Computers** table, select option **2**.
11. Click **OK** and run the query. The **Pentio** record is now shown, but because it has not been repaired, the last three fields are blank.

Serial Num	Model	Job No	Engineer	Date
B12345A	Pentio			
C44477	C4	2	Stephen	06/04/2008
C4535F	C4	4	David	22/04/2008

12. Save the query as **Outer join** and close it.
13. Reopen the **Outer join** query in **Design View**. Add the criteria "**Is Null**" to the **Job No** field.
14. Run the query. Only the **Pentio** record is displayed, because this is the only record without a **Job No**. This is known as a **subtract join** and displays only computers that have no repair.



*The actual link is still an outer join, the **Is Null** criteria makes it a subtract join.*

15. Save the query as **Subtract join** and close it. Close the database.

Driving Lesson 28 - Self Joins

Park and Read

A **self join** describes the situation where a table needs to be linked to itself. For example, in a **Staff** table, with **Employee Number** as the primary key, there may be a field showing who each employee's manager is, using the manager's employee number. This employee number will already appear as the primary key for a record in the table (managers are also employees). To display the manager's name instead of their number in a query, a link to a copy of the **Staff** table will need to be set up, using **Manager** as the link field.

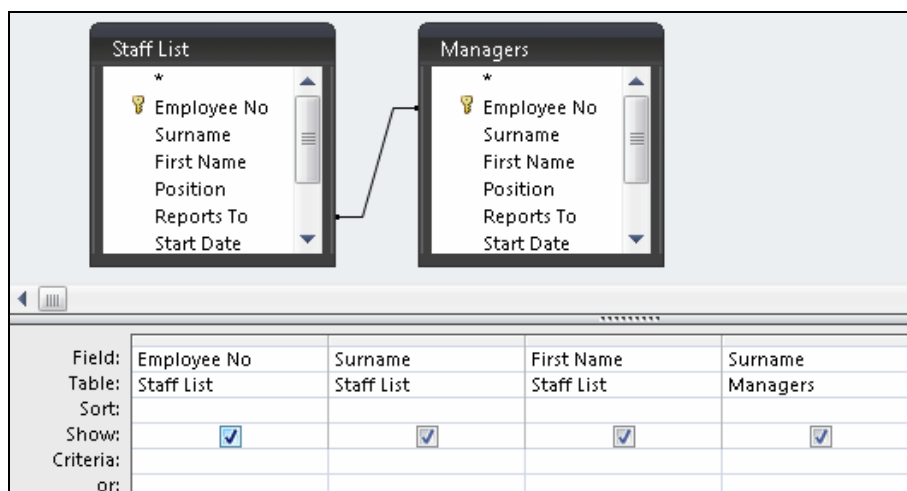
Manoeuvres

1. Open the **Staff** database and create a new query in **Design View**. At the moment, the manager that each member of staff reports to is only shown in the **Reports To** field as a staff number. A query can be created to display the manager's name, but first a **self join** must be created.
2. To create the self join, add the **Staff List** table to the query grid twice.




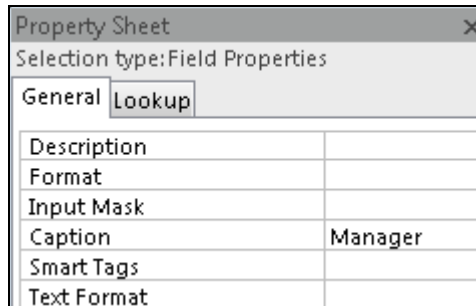
The table itself is not duplicated; it is just referred to by two different names.

3. The second list is given a different name, **Staff List_1**. To change this, right click on the list and select **Properties** from the shortcut menu.
4. Change the **Alias** entry to **Managers** and close the **Property Sheet**.
5. Drag the **Reports To** field from **Staff List** to the **Employee No** field in the **Managers** to create the link.
6. From **Staff List**, add **Employee No**, **Surname** and **First Name** to the grid and from **Managers**, add **Surname** to the grid again.



Driving Lesson 28 - Continued

- To change the last field title so that it displays **Manager** instead of **Surname**, click anywhere in the last field in the query grid to select it, then click  **Property Sheet** from the **Show/Hide** area of the **Design** tab.
- Change the **Caption** property to **Manager**.



- Close the **Property Sheet** and run the query.

Employee No	Staff List.Surname	First Name	Manager
100	Nyder	Peter	Cheshunt
101	Cheshunt	Richard	Singh
112	Myers	Anthony	Singh
301	Lee	Ciara	Myers
321	Chapman	Ian	Cheshunt
403	Morris	Tracy	Cheshunt
500	Valdron	Brian	Myers
505	Kline	William	Cheshunt
509	Jones	Lesley	Cheshunt
536	Singh	Vikram	Singh
688	Chamberlain	Anthony	Cheshunt
704	Parr	Norma	Cheshunt
1000	Ripley	Ellen	Myers

- Notice how the manager's name and not their employee number appears in the **Manager** field. This is because the self join created earlier causes the query to look up the corresponding name for each employee number entered in the **Reports To** field and return that value.



*The actual link in a self join is still either an inner or outer join, it is the fact that it links to a copy of the same table that makes it a self join. Like any relationship, this can be deleted or modified by right clicking on the connecting line and selecting **Delete** or **Edit Relationship**.*

- Save the query as **Self join** and close it.
- Close the database.

Driving Lesson 29 - Referential Integrity

Park and Read

Referential Integrity is a set of rules which can be applied to relationships, ensuring they are valid and that data is not accidentally deleted or changed. It may be applied when specific conditions are met: the matching field from the primary table is a primary key, the related fields are the same data types and both tables belong to the same database.

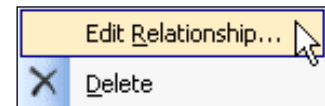
Enforcing referential integrity controls the updating of primary key data in the primary table and the deletion of any record from the primary table, if a related record exists elsewhere. A record cannot be added to a related table if a record does not exist in the primary table, e.g. there can be no repair record for a computer unless an associated computer record exists in the primary table.



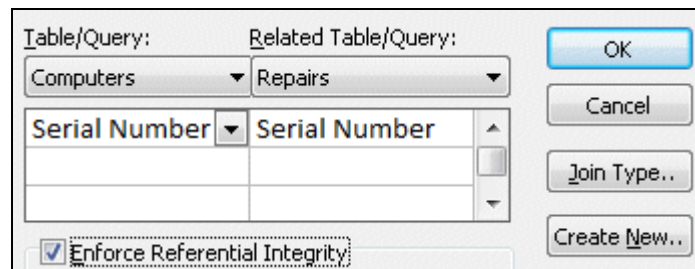
Manoeuvres

1. Open the **Custom Computers** database and click on the **Relationships** button, to display the relationships.

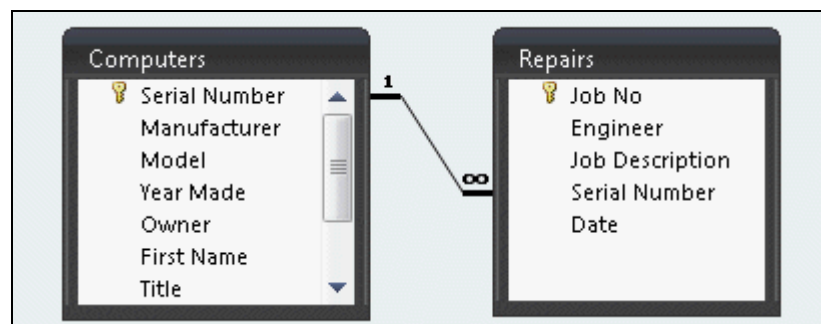
2. Right click with the mouse on the relationship line between **Computers** and **Repairs** and select **Edit Relationship** from the menu.



3. In the **Edit Relationship** dialog box, check the box for **Enforce Referential Integrity**.



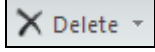
4. Click **OK**.

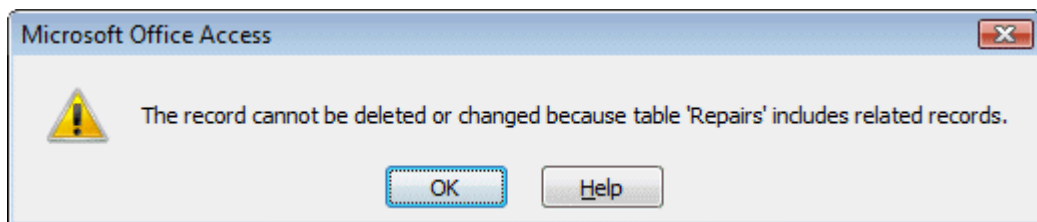


Driving Lesson 29 - Continued



Enforcing referential integrity will change the relationship line to show the type of relationship, in this case, one to many.

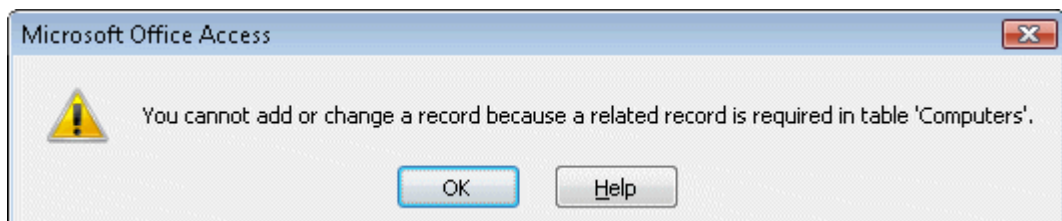
5. Close the **Relationships** window.
6. Open the **Computers** table in **Datasheet View**.
7. Click to select the whole record **T2457** and click  from the **Records** group on the **Home** tab. The record cannot be deleted, as there are related records in the **Repairs** table.



8. Click **OK**. Close the table.
9. Open the **Repairs** table in **Datasheet View**.
10. Click in the empty record row at the end of the table and enter the following information:

18, David, Renew Case, XY33, 29/06/2008

11. Press **<Enter>** after the last entry. As there is no serial number for this job in the **Computers** table, a new job record cannot be created.



12. Click **OK**. Delete the information just entered using **Undo**.
13. Close the table and leave the database open for the next Driving Lesson

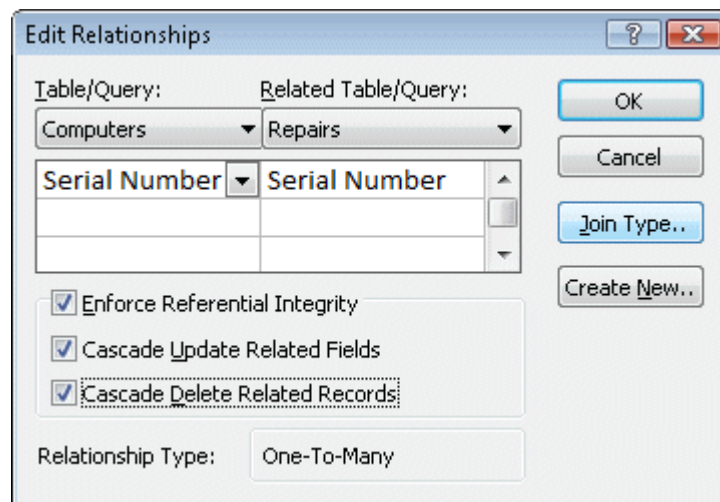
Driving Lesson 30 - Cascade Options

Park and Read

Enforcing referential integrity prevents the deletion or amending of records that would cause related tables to lose integrity. If **Cascade** options are set however, such deletions and amendments are allowed, but related tables will be automatically adjusted. Records in related tables will be automatically deleted or amended.

Manoeuvres

1. The **Custom Computers** database should be open from the previous Driving Lesson.
2. To allow editing or deletion of records, the relationship between the tables must be changed. Click on the **Relationships** button.
3. Right click on the linking line between **Computers** and **Repairs** and select **Edit Relationship**.
4. Edit the relationship as follows: check the boxes for **Cascade Update Related Fields** and **Cascade Delete Related Records**.

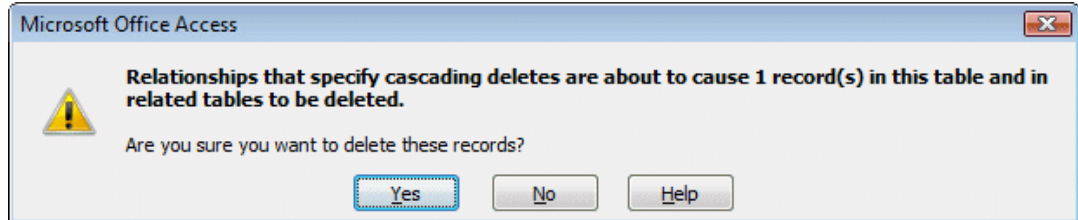


5. Click **OK**. This means that any changes made in one related table will be reflected in the others.
6. Close the **Relationships** window.
7. Open the **Computers** table in **Datasheet View** and delete the record for the **Registry Gigathon, R65488G**. This computer has a repair record on the **Repairs** table, record **15**.



Driving Lesson 30 - Continued

8. Deletion is now allowed but the **Cascade Delete Related Records** option means that the related records (in the **Repairs** table) will also be deleted.



9. Select **Yes** to continue.



*Clicking **No** would cancel the original deletion.*

10. In the **Computers** table, change the serial number **T2457** to **T5724**.
11. Close the table and open the **Repairs** table in **Datasheet View**. The **Cascade Delete Related Records** option has caused the job for the **Registry Gigathon** (job 15) to be automatically removed.
12. Look at the jobs associated with the **Triton** computer (jobs 1, 7 and 14). The **Cascade Update Related Records** option has caused the serial number for these jobs to be automatically amended.
13. Close the table and the database.

Driving Lesson 31 - Revision

This is not an ECDL test. Testing may only be carried out through certified ECDL test centres. This covers the features introduced in this section. Try not to refer to the preceding Driving Lessons while completing it.

1. Open the **Organiser** database.
2. In the **Meetings** table add a **Primary Key** to the **Meeting ID** field.
3. In the **People** table add a **Primary Key** to the **Contact ID** field.
4. Create a **One to Many** relationship between the tables using the **Contact ID** field.
5. Enforce **Referential Integrity**, with no **Cascade** options.
6. Display the **People** table. Use the subdatasheet view to see how many meetings **Ian Chapman** has recorded.
7. On the same display change the **Contact ID** for **James Tebb** to **KT**. Try to update the record. Why is the update not allowed?
8. Close all tables, redisplay the relationship between the tables and enforce **Cascade Update Fields** and **Delete Related Records**.
9. Change the **Contact ID** for **James Tebb** to **KT** again. Is the update allowed now?
10. In the **Meetings** table change the **Contact ID** field in record 5 from **GS** to **AS** and try to update the record. Why is it still not allowed?
11. Undo the last key stroke then close the table without saving any changes.
12. Close the database.



Check the answers at the back of the guide.

If you experienced any difficulty completing the Revision, refer back to the Driving Lessons in this section. Then redo the Revision.

Driving Lesson 32 - Revision

This is not an ECDL test. Testing may only be carried out through certified ECDL test centres. This covers the features introduced in this section. Try not to refer to the preceding Driving Lessons while completing it.

1. Open the database **Brief Encounter**, a database for an introduction agency.
2. Open the table **Personal records** and scroll to the field **Referred by**. This uses the **BE Number** to indicate who might have referred this client.
3. Design a query that will show the name, rather than the number, of the person referring the client.
4. Show the **First Name** and **Surname** of the client and the **Surname** of the referee.
5. Change the **Caption** of this last field to **Personally referred by** and apply a **Sort** (ascending) to it.
6. Save the query as **Referrals**.
7. Use the same database to create a query showing which men have had introductions.
8. Display the **Surname** and **First Name** fields from the **Personal Records** table and the **Date of Encounter** and **Dating Number** from the **Encounters** table. Include, without displaying, any other fields that may be necessary.
9. Make sure the result will only show men.
10. Check that the join type is **inner** and run the query.
11. How many introductions are there?
12. Save the query as **Introduction**.
13. Use the same query grid to show all introductions by **Surname**, regardless of gender and include those surnames that are yet to receive an introduction (change the type of join). How many records are shown?
14. Save the query as **Dates** then close the query and the database.



Check the answers at the back of the guide.

If you experienced any difficulty completing the Revision, refer back to the Driving Lessons in this section. Then redo the Revision.

Once you are confident with the features, complete the Record of Achievement Matrix referring to the section at the end of the guide. Only when competent move on to the next Section.