Skill Set 2

Creating and Linking Tables

By the end of this Skill Set you should be able to:

Create Databases
Create Tables
Understand Data Types
Apply Primary Keys
Apply and Modify Relationships
Understand Different Relationship Types
Understand Referential Integrity
Create Linked Tables

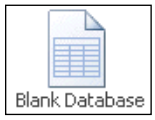
Exercise 9 - Creating a New Database

Knowledge:

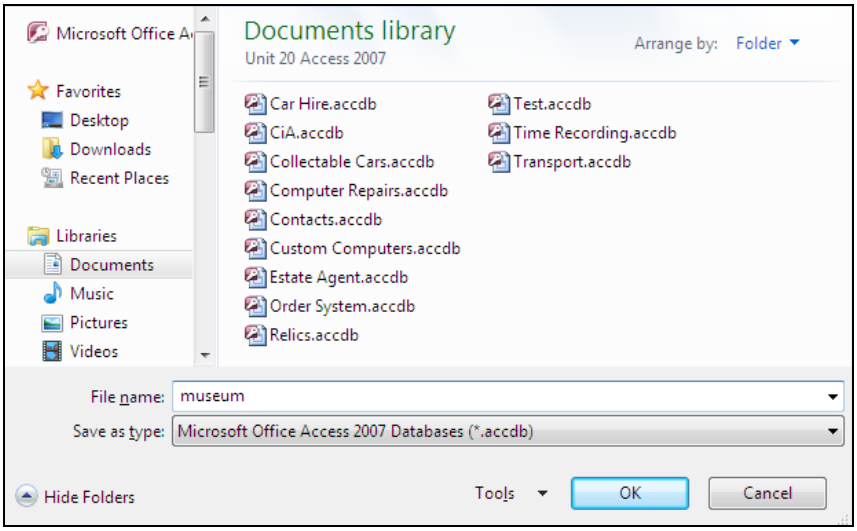
Unlike most applications, *Access* creates and saves a blank database before any objects have been created or any data has been added.

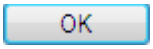
Activity:

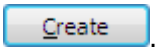
1. Start *Access*. From the *Access* **Getting Started** screen, click **Blank Database**. This displays a new panel on the right of the screen.



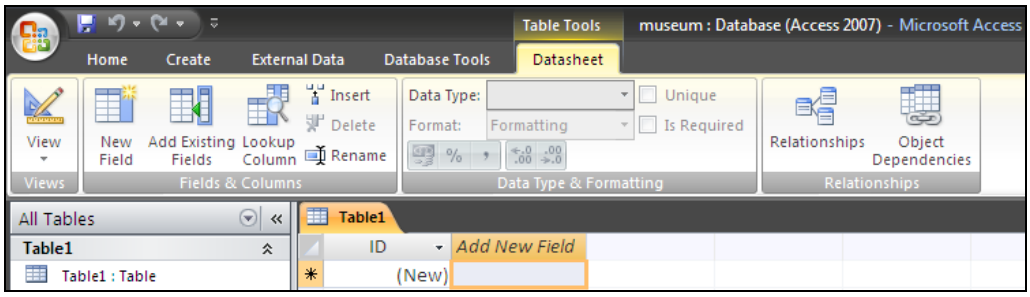
2. Use the  button to browse for a location to put your database. Make sure the **Folders** pane is shown on the left and use it to locate the supplied data folder for this guide (see page 3). Enter a **File name** of **museum**.



3. Click  in the **File New Database** dialog box.

4. In the **Blank Database** panel, click  .

5. The **Access Database Window** is now displayed, with the database name (**museum**) in the **Title Bar** and a default empty table **Table1** open.



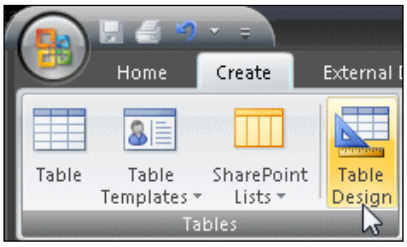
6. In order to demonstrate a more general method of creating tables, close the default table, by right clicking on the **Table1** tab, and selecting **Close** from the shortcut menu. Leave the database open.

Exercise 10 - Creating Tables

Knowledge:

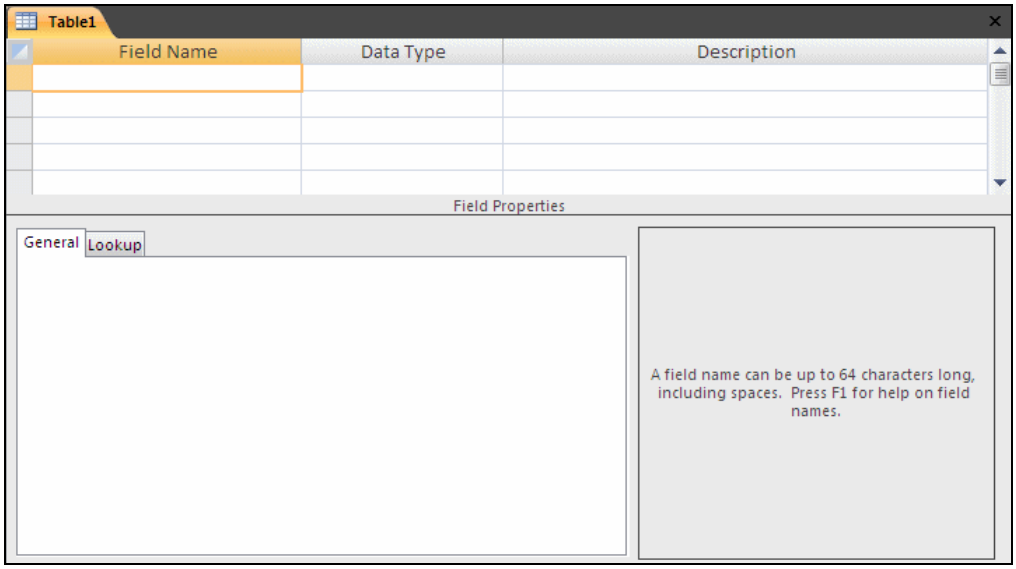
Tables are the fundamental objects of the database. However complex the database, all data is stored in tables. Tables can be created using a wizard which gives step by step instructions for the initial setting up and applies default formatting, such as date formats and number formats. Alternatively, a table can be created in **Design View** which allows a free hand for formatting.

Activity:



1. In the new **museum** database, display the **Create** tab and click the **Table Design** button. A new table appears in the work area in **Design View**, which allows the structure of the table to be defined.

2. Notice that the window is divided into two areas. The upper half of the **Design View** screen allows fields for the table to be defined. The lower half has panels to specify more detailed properties of each field, as well as some explanatory text, specific to the property being defined.



3. In the first **Field Name** row, enter **Artefact**.

4. Leave the **Design** window open for the next exercise.

Exercise 11 - Data Types

Knowledge:

Each field in an *Access* table has a **Data Type** associated with it. This defines the type of data that can be contained within the field. Some of the available field types are as follows:

Type	Description
Text	Any combination of characters and numbers up to a maximum length of 255. The default length is 50.
Memo	Same content as Text but up to 64,000 characters allowed.
Number	Numeric data only. Essential for calculations.
Date/Time	Will only accept dates or times. A variety of formats may be specified.
Currency	Numeric data related to currency. The currency symbol can be changed.
AutoNumber	A sequential number applied by *Access* to each new record in a table.
Yes/No	A logical field that represents yes/no or true/false values. Actually contains either a 1 or 0.
Hyperlink	Hyperlinks to other objects. Not covered in this unit.
Lookup Wizard	A field which allows values to be chosen from a list or table. Lookup fields are defined for Forms in a later Skill Set.

Try to think ahead when allocating field types. An item code may consist of all numbers, but you are unlikely to want to do calculations with it, but one day you might want to add characters, e.g. **87654321A**, so make it **Text**.

Activity:

1. In the table **Design** window open from the previous exercise, click in **Data Type** for the first field. The default type of **Text** is shown.

2. Press <**Tab**>. The cursor is now in the **Description** column; this column is optional. Enter **Name of Artefact** into this column, then press <**Tab**>.

3. Enter a second row with a **Field Name** of **Era**, **Data Type** of **Text** and **Description** of **Historical Period**.

4. For the next **Field Name**, enter **Date Acquired** and press <**Tab**>.

5. Click the **Data Type** drop down arrow and select **Date/Time** from the list.

6. Press <**Tab**> to move into the **Description** column and type **Date the artefact was bought** then press <**Tab**> again.

7. Enter **Cost** as the fourth field name, <**Tab**>, then select **Currency** as the **Data Type** and enter **Buying price** as the **Description**.

Exercise 11 - Continued

8. The next **Field Name** is **Quantity**, the **Data Type** is **Number** and the **Description** is **Number of artefacts bought**.

9. For the next field enter **Supplier Name**, the **Data Type** is **Text** and the **Description** is **Name of Supplier**.

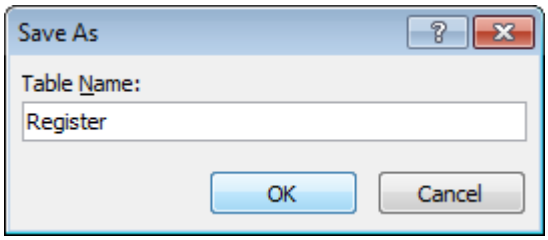
10. For the next field enter **Supplier Address**, the **Data Type** is **Text** and the **Description** is **Address of Supplier**.

11. For the next field enter **Information**, the **Data Type** is **Memo** and the **Description** is **History**.

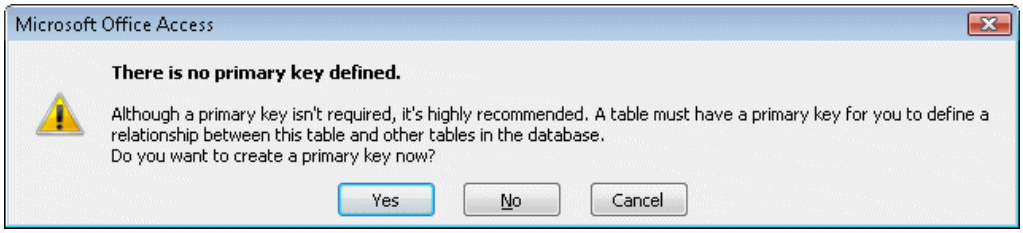
12. It is decided to give each record a unique reference number. Click in the first field name (**Artefact**) and select **Insert | Rows** from the **Design** tab.

13. A blank row is inserted. Enter a **Field Name** of **Artefact Reference**, select a **Data Type** of **AutoNumber**, and enter a **Description** of **Artefact Record ID**.

14. Right click on the **Table1** tab and click **Save**. Because the table has not been saved before, the **Save As** dialog box is displayed. Type **Register** as the **Table Name**.



15. Click **OK**. A message is displayed regarding **Primary Keys**.



16. Although not essential for a single table database, click **Yes**. The **AutoNumber** field will be automatically selected to be the **Primary Key**.

Note:	*At present this is not an efficient database design. As the same suppliers provide different artefacts, it is inefficient to store the supplier name and address on each artefact record. It would be better to create another table containing supplier details and relate the relevant information to each artefact. This will be done in a later exercise.*

17. Close the table and the **museum** database.

Exercise 12 - Applying a Primary Key

Knowledge:

When creating 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 can be identified by creating a field which contains unique data, e.g. a serial number or identification number. This is called a **Primary Key**.

If there is no suitable field in the table, a new field can be added specifically.

As well as enabling the linking of tables, use of a **Primary Key** prevents duplication of records in a table (because it is unique and required) and because it is also automatically an **Index**, it also allows sorting and querying to be performed more efficiently.

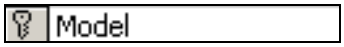
Other fields can be defined as **Indexes** without being primary keys. An **Index** is an automatically maintained sorting sequence for a table which can speed up searches based on a particular field. Other fields which could also be used to identify a record are sometimes called **Secondary Keys**.

Activity:

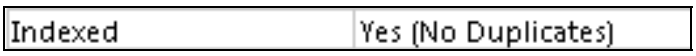
1. Open the **Custom Computers** database. This contains a table with details of computers, including serial number, and a table of repair details which also contains serial number.

2. Open the **Computers** table in **Design View** and look at the fields with a view to selecting a field to become the **Primary Key**.

3. Click in the **Model** field. Click the **Primary Key** button on the toolbar. A **Primary Key** is then applied to the **Model** field.



4. Look at the **Field Properties** for this field.



5. Notice in the field properties that the **Indexed** property is automatically set to **Yes (No Duplicates)**. This means that there cannot be two records in the table with the same Model. 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.

6. Click in the **Serial Number** field and click on the **Primary Key** button again.

7. **Serial Number** is now designated as the **Primary Key** field. This is an ideal choice, as a serial number is a unique identifier for any particular computer.

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

Exercise 12 - Continued

8. Save and close the table.

9. Open the **Repairs** table in **Design View**. A **Primary Key** for this table has already been applied to the **Job No** field.

> ***Note:*** ***Serial Number*** *is not a suitable* ***Primary Key*** *for the* ***Repairs*** *table as there could be more than one job record for the same computer.*

10. Click the **Indexes** button on the **Design** tab, to view the indexes applied to this table. An index in a table automatically maintains a sort sequence for the records in the table. The **Primary Key** field is automatically an index.

> ***Note:*** *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.*

> ***Note:*** ***Primary Key*** *is a special example of an* ***Index****. Other fields can be defined as* ***Indexes*** *and can, if appropriate, also be defined as* ***Unique*** *and* ***Not Null****. These are sometimes called* ***Secondary Keys*** *and can help to maintain data integrity.*

11. Click in the **Unique** box under **Index Properties**, click the drop down arrow and select **No** from the list.

12. A message box explains why this change is not allowed. Read the text and click **OK**. All **Index Properties** for a **Primary Key** field are fixed.

13. Close the **Indexes** dialog box and the **Repairs** table.

14. Leave the **Custom Computers** database open.

Exercise 13 - Applying Relationships

Knowledge:

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 applied between tables which contain a common field. Usually, the related field in the first table, containing the unique record, is the **Primary Key**. The related field in the second table, which is used to link to information from the first table, is known as the **Foreign Key**.

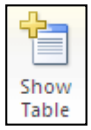
Applying relationships allows many smaller tables to be linked together to form the complete database, improving its overall efficiency. In this exercise then, if the **Computer** and **Repairs** tables are linked using the common field serial number, there is no need to have all computer details on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

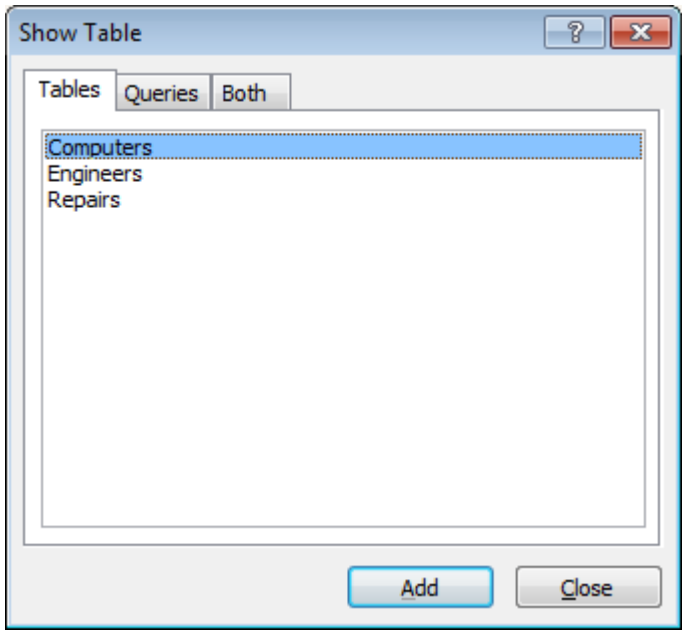
There are different types of relationships, introduced over the next few exercises.

Activity:

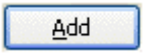
1. With the **Custom Computers** database open, display the **Database Tools** tab and click the **Relationships** button. A blank relationship area is displayed and the **Design** tab is shown on the **Ribbon**.

2. If the **Show Table** dialog box does not appear, click the **Show Table** button on the **Design** tab.

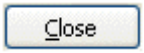




Exercise 13 - Continued

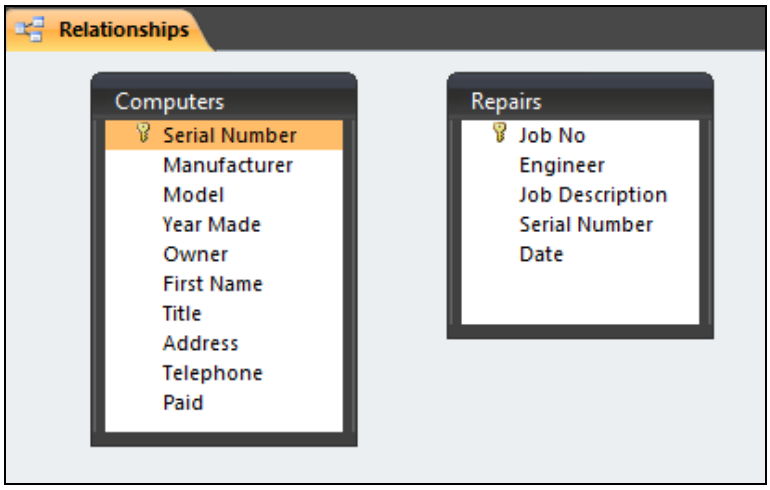
3. With the **Computers** table highlighted, click 

to place the table in the **Relationships** window.

4. Click on the **Repairs** table and again add it to the window, then click 

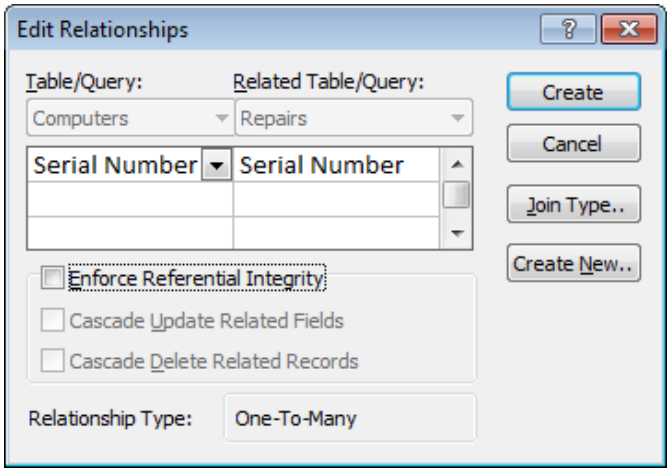
to remove the **Show Table** dialog box.

5. Resize the table boxes to see all of their fields. Notice the **Primary Keys** for each table are indicated with a **Key** symbol.

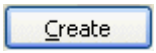


6. Highlight the **Serial Number** field in the **Computers** table.

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.



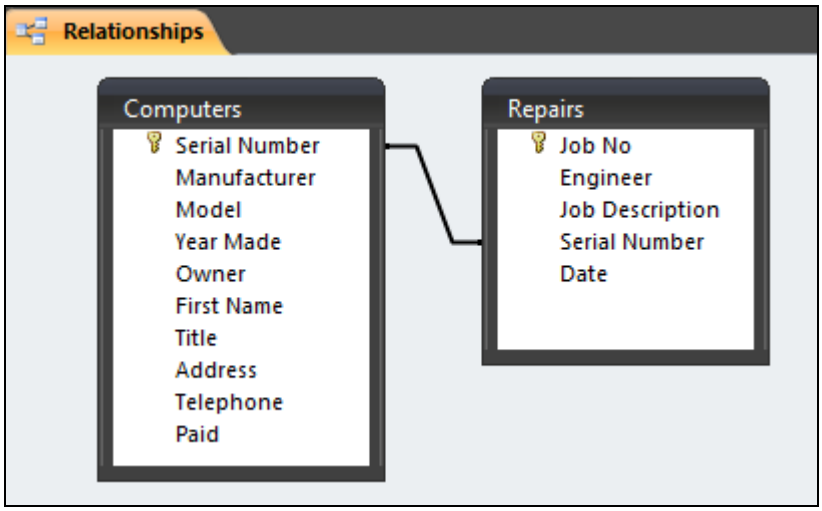
8. In the **Edit Relationships** dialog box, note the **Relationship Type** is **One-To-Many** (one computer record can be linked to many repair records). This is the most common type of relationship.

9. Click 

to create the relationship.

Exercise 13 - Continued

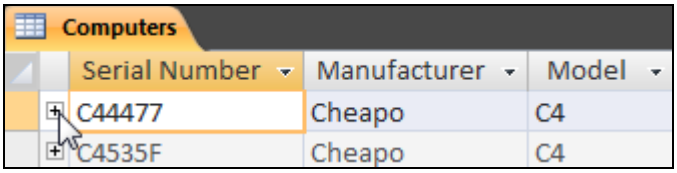
10. Notice how the relationship between the tables is now symbolised by a line, linking the same field, **Serial Number**.



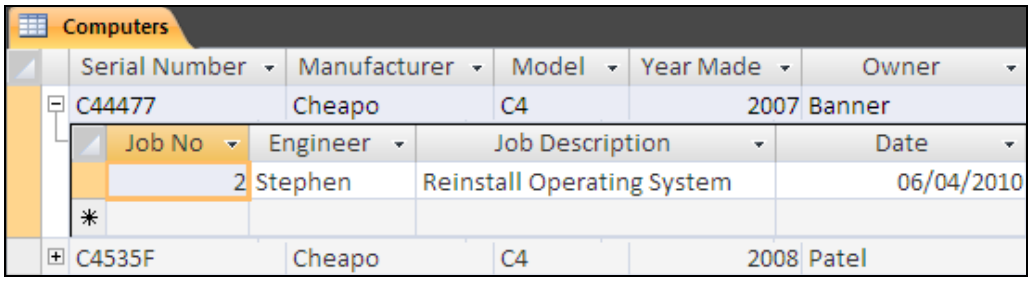
Note:	*The relationship is **One-To-Many**. This means that one record from **Computers** can have many related records in **Repairs**, i.e. one computer may have many repairs. The fact that one of the fields in the link is a primary key and the other is not, defines the relationship as **One-To-Many**.*

11. Right click in a blank part of the **Relationships** window and select **Close** to close it. Select **Yes** when prompted to save the changes to the relationship.

12. Open the **Computers** table. As this table is now linked to the **Repairs** table, a subdatasheet button appears next to each **Computer** record.



13. Click the expand subdatasheet button next to the first record. The related **Repairs** record(s) for this computer are shown – notice that **Serial Number**, the related field in the second table (the foreign key), is not shown again.



Note:	*Click the subdatasheet button again to hide the **Repairs** record(s).*

14. Close the table and leave the database open for the next exercise.

Exercise 14 - One to One Relationships

Knowledge:

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 table with many fields or if part of a table is removed for security reasons or if the second table contains optional data.

Activity:

1. To create a **One-to-One** relationship, using the **Custom Computers** database, first use the **Table Design** button on the **Create** tab to create a table in **Design View**.

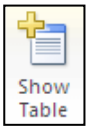
> ***Note:*** *Do not create the table in **Datasheet View** as **ID** fields will be automatically added.*

2. Add two fields, **Job No** and **Charge**. 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**.

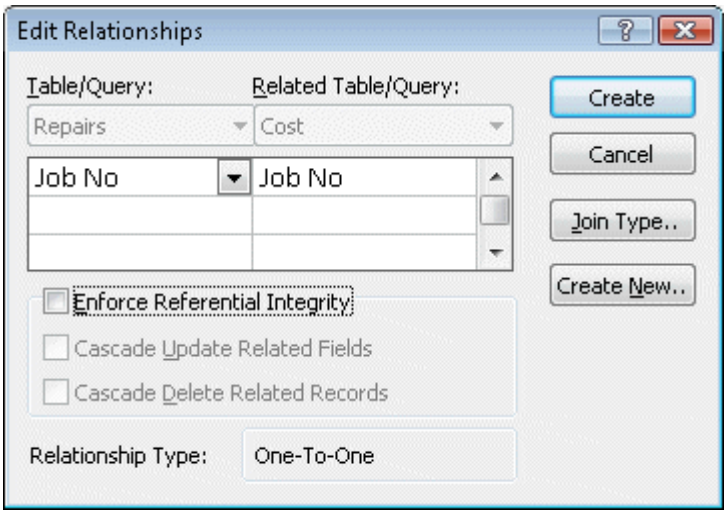
4. Save the table as **Cost**. Close it without adding any data.

5. Open the **Relationships** window showing the existing relationship and use the **Show Table** button to display the dialog box.



6. Select **Cost** then click **Add** to add it to the **Relationships** window. Close the **Show Table** box.

7. 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, the **Relationship Type** is displayed as **One-To-One** in the **Edit Relationships** box.

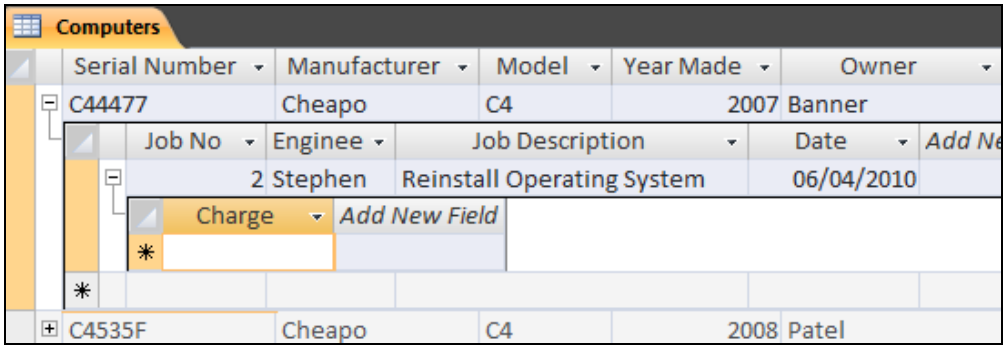


Exercise 14 - Continued

8. Click **Create** and close the **Relationships** window, saving at the prompt.

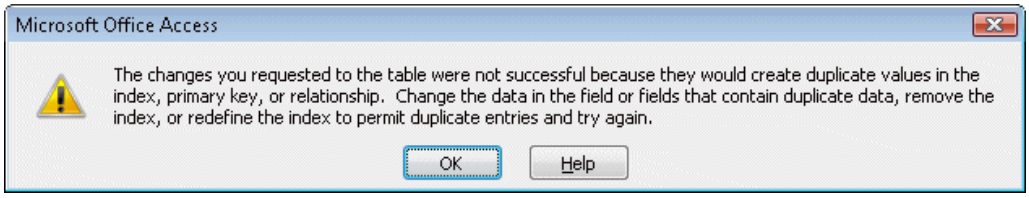
9. Open the **Computers** table and click the expand subdatasheet button next to the first record.

10. The related **Repairs** records are shown. There is also now an expand subdatasheet button in the **Repairs** table. Click on it. The **Cost** records are shown for this **Job**.



11. At the moment there is no data in the **Charge** field. Type **250** in the **Charge** field for this repair and press <**Enter**>.

12. Try adding another charge for this repair. Remember there is a **One-to-One** relationship in force, so only one charge should be allowed. A warning dialog box is displayed as below.



13. Click **OK** once you have read it. To get rid of this useless information is to click the **Undo** button,  from the **Quick Access Toolbar** or press <**Esc**>.

14. Open the subdatasheet for each job in turn and add suitable **Charges**.

Note:	*Some computers will have more than one associated job record, but no job can have more than one cost record.*

15. Close the table and the database.

Exercise 15 - Many to Many Relationships

Knowledge:

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.

Activity:

1. Open the **CiA** database. To create a **Many-to-Many** relationship between the **Orders** table and the **Products** table (an order can include several products and a product can be on several orders) it is first necessary to have an intermediate, junction table.

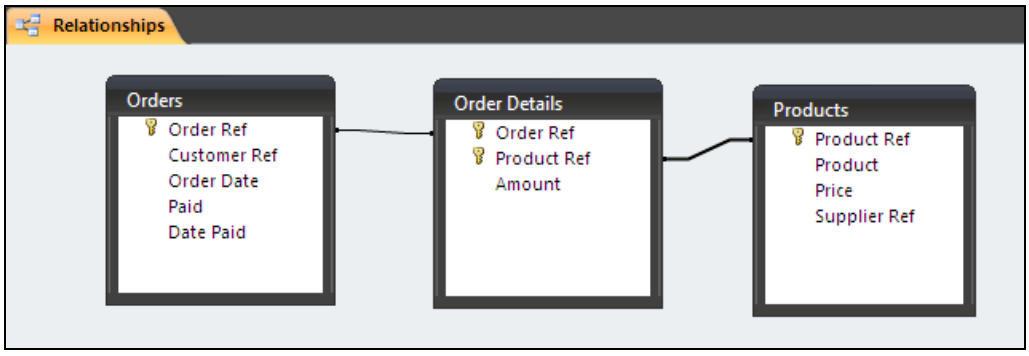
2. The table needs to contain the fields **Order Ref** (**Number**) and **Product Ref** (**Text**), and the **primary key** for this new table will be defined as the combination of both fields (neither individual field will be unique on the table). This table already exists in the database. Open the **Order Details** table in **Design View**.

Field Name	Data Type
Order Ref	Number
Product Ref	Text

3. View the **Relationships** window and place the **Orders**, **Order Details** and **Products** tables on to it.

4. Create a **One-to-Many** relationship between the **Orders** and **Order Details** tables using the **Order Ref** field.

5. Create a **One-to-Many** relationship between the **Products** and **Order Details** tables using the **Product Ref** field. The overall effect is that now a **Many-to-Many** relationship exists between the **Orders** and **Products** tables.



6. Close the **Relationships** window, saving at the prompt, then close the database.

Exercise 16 - Referential Integrity

Knowledge:

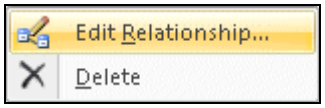
Referential Integrity is a set of rules which can be applied to relationships, ensuring they are valid and that data is not accidentally deleted or changed. It may be applied when specific conditions are met: the matching field from the primary table is a primary key, the related fields are the same 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 job record without an associated computer record in the primary table.

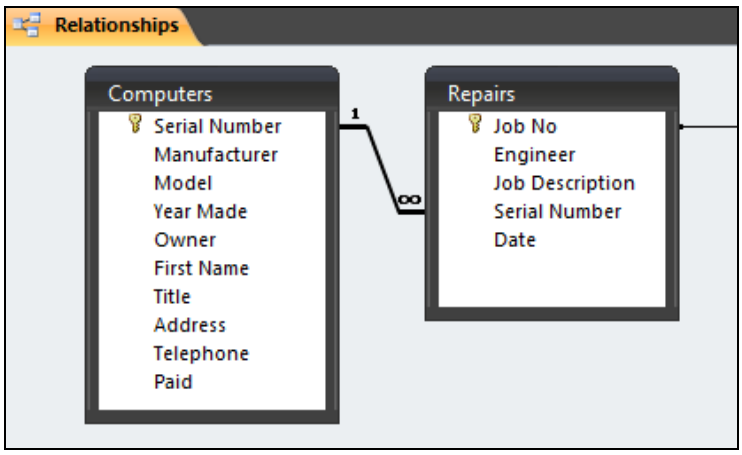
Activity:

1. Open the database **Custom Computers**, display the **Database Tools** tab and click the **Relationships** button.

2. Right click with the mouse on the relationship line between **Computers** and **Repairs** and select **Edit Relationship** from the menu (notice there is also an option to delete the relationship here).



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



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

4. Close the **Relationships** window. If a save prompt is shown, click **Yes**.

5. Leave the database open for the next exercise.

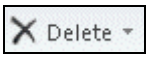
Exercise 17 - Cascade Options

Knowledge:

If referential integrity is enforced, deleting records from the primary table or changing a primary key is either prevented or controlled. Similarly, a record in a related table may not be created, if a matching record is not available in the primary table.

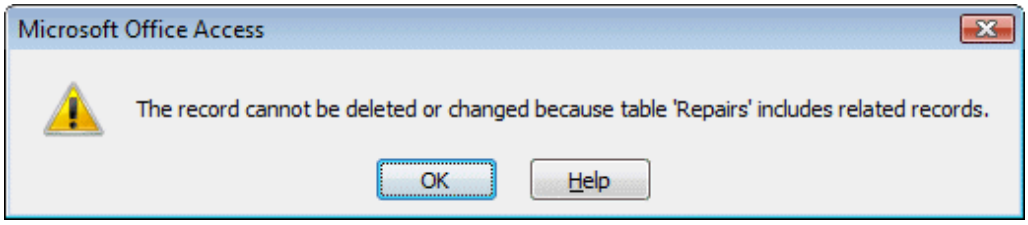
Activity:

1. In the **Custom Computers** database, open the **Computers** table in **Datasheet View**.

2. Click in the record **C44477** and click the drop down arrow on the **Delete** button, 

from the **Records** group on the **Home** tab.

3. Select **Delete Record**. The record cannot be deleted, as there are related records in the **Repairs** table.



4. Click **OK**. Close the table.

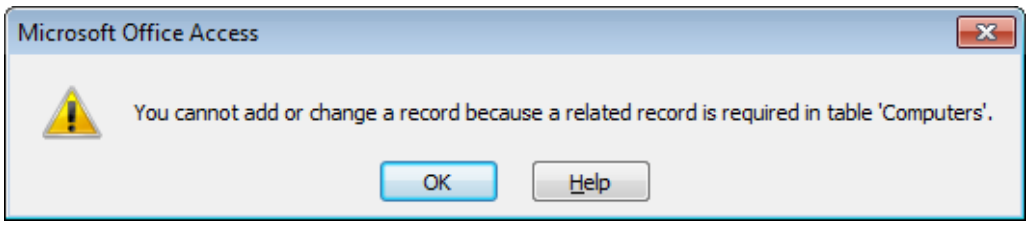
5. Open the **Repairs** table in **Datasheet View**.

6. Click the **New Record** button, 

from the **Home** tab and enter the following information:

22 David Maintenance T1234 04/01/10

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



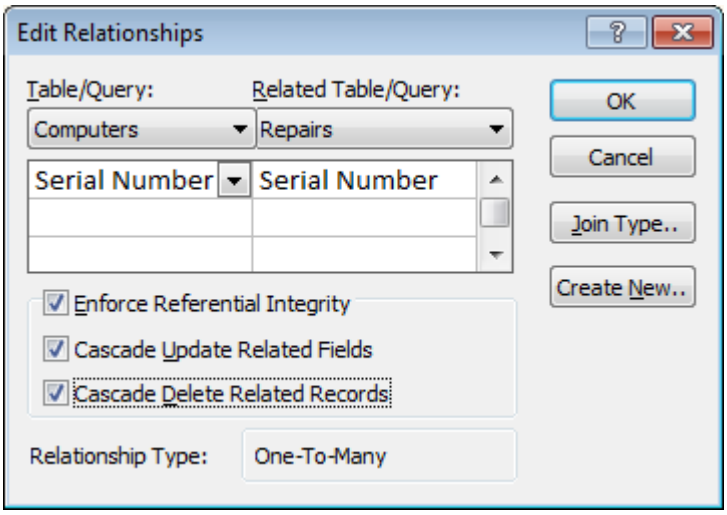
8. Click **OK**. Delete the information just entered using **Undo** or <**Esc**>. Close the table and return to the **Database Window**.

9. To allow editing or deletion of records, the relationship between the tables must be changed. Click the **Relationships** button.

Exercise 17 - Continued

10. Right click on the linking line between **Computers** and **Repairs** and select **Edit Relationship**.

11. Edit the relationship as follows: check the boxes for **Cascade Update Related Fields** and **Cascade Delete Related Records**.

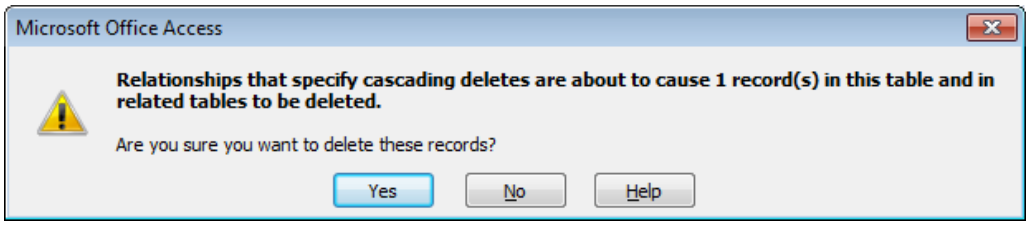


12. Click **OK**. This means that any changes made in one related table will be reflected in the others.

13. Close the **Relationships** window.

14. Open the **Computers** table in **Datasheet View** and click the subdatasheet button for computer **R65488G**. There is a repair record (record **15**) for this computer on the related table, **Repairs**.

15. Delete the record for computer **R65488G**. As the **Cascade** options have been set, deletion is now allowed, but there is a warning that records in related tables will also be deleted.



16. Click **Yes** to continue.

17. In the **Computers** table, change the serial number **C7689E** to **C7689G**.

18. Close the table and open the **Repairs** table in **Datasheet View**. The job for computer **R65488G** (job 15) is removed and the serial number change has been reflected in the table (job 13).

19. Close the table and the database.

Exercise 18 - Creating Linked Tables

Knowledge:

Linking tables that already contain data can cause problems if the existing data is not consistent across all the tables. Normally, tables are linked as they are being designed, before any data is added. Integrity of the data can then be controlled by the relationships.

Activity:

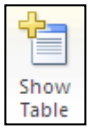
1. Open the **museum** database created earlier, and open the **Register** table in **Design View**.

2. Delete the **Supplier Name** and **Supplier Address** fields and replace them with a field named **Supplier Code**. The **Data Type** is **Text** and the **Description** is **Supplier Reference Code**.

3. Save and close the **Register** table and create a new table in **Design View**. Add the following fields: **Supplier Code**, **Supplier Name**, **Address**, **Contact**, **Telephone No**, all **Data Types** to be **Text**.

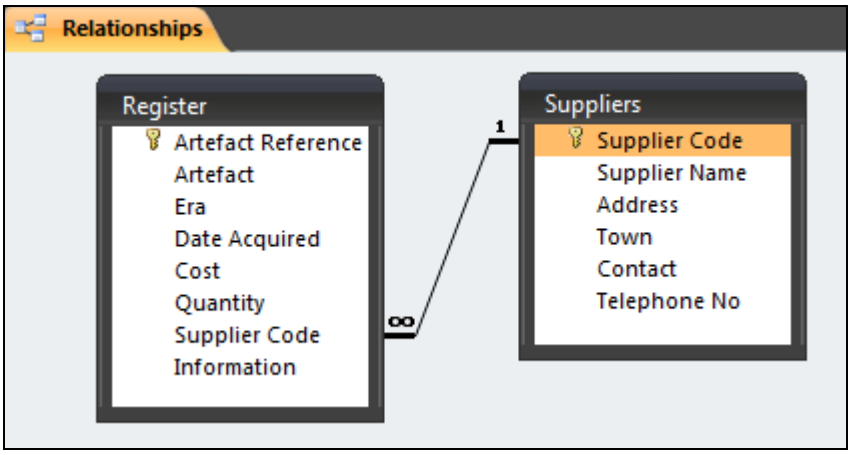
4. Apply **Primary Key** to the **Supplier Code** field, then save the table as **Suppliers** and close it.

5. Open the **Relationships** window and if necessary, use the **Show Table** button to display the dialog box.



6. Add both **Register** and **Suppliers** to the **Relationships** window. Close the **Show Table** box.

7. Create a **One-To-Many** relationship between the **Register** table and the **Suppliers** table using the **Supplier Code** field. Select **Referential Integrity** and all **Cascade** options and click **Create**.



8. Now more information can be held for each supplier in the **Suppliers** table but all that is held on the **Register** table is the field **Supplier Code**.

9. Save and close the **Relationships** window and close the database.

Exercise 19 - Develop Your Skills

You will find a *Develop Your Skills* exercise at the end of each Skill Set. Work through it to ensure you've understood the previous exercises.

In the Develop Your Skills exercises, a sample task will be developed which could demonstrate the assessment criteria. The sample task is to develop a database to enter, monitor and analyse the time spent by company staff across a range of projects. The information will come from the completion of weekly time sheets.

Tables will be required to hold data on **Staff Members**, **Time Sheets** and **Projects**. Normalisation will mean that the time sheet data will need to be stored in two tables, **Time Sheet Header** and **Time Sheet Lines**.

A **Time Sheet Header** will identify a staff member and contain a week number. It will be linked to many possible **Time Sheet Lines**. A **Time Sheet Lines** record will identify a **Header** record and a project and the time spent on that project.

The **Staff** and **Project** tables will hold fixed data for those entities.

Data analysis will be possible by person (which projects has a certain person been working on?), or by project (which people have been working on a specific project?). Many summaries will be possible (what is the total time spent on a project?). If the staff table includes department and the project table includes project type, then analysis can be broadened, e.g. how much time does the training department spend on consultancy type projects?

Activity:

1. Create a new database and save it as **Time Sheet**.

2. Create a table containing the following fields and save it as **Staff**:

Field Name	Type	Description
Staff No	Text	Primary Key
First Name	Text	
Surname	Text	
Department	Text	
Date of Birth	Date/Time	
Cost	Currency	Calculated cost
Extension	Text	Internal telephone number

Note: *For all tables, accept the default properties for every field.*

3. Create a table containing the following fields and save it as **Project**:

Field Name	Type	Description
Project Code	Text	Primary Key
Project Name	Text	
Project Type	Text	
Active	Yes/No	

Exercise 19 - Continued

4. Create a table with the following fields and save it as **Time Sheet Header**:

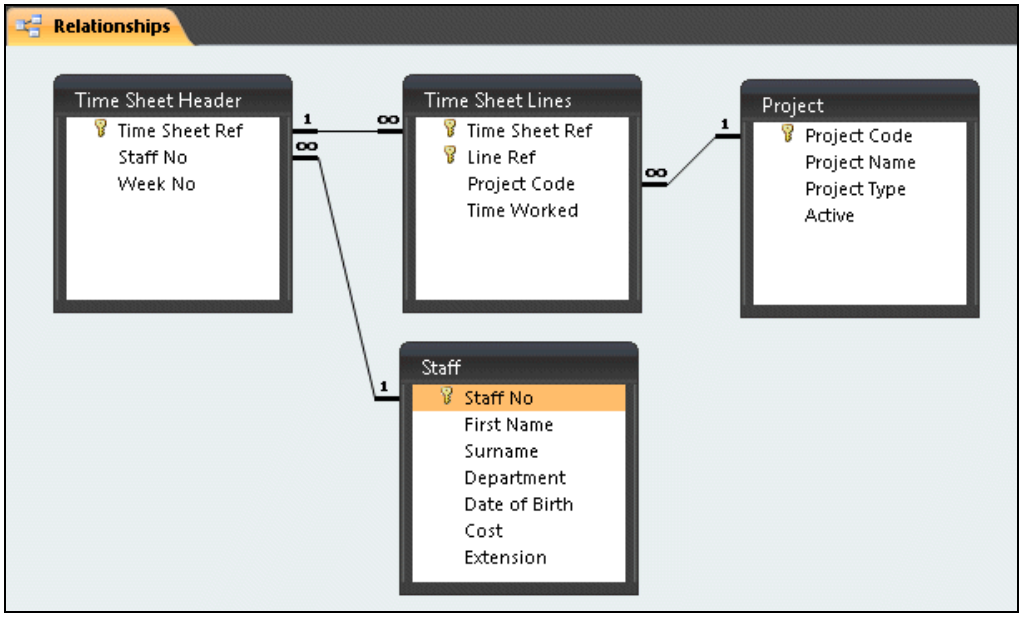
Field Name	Type	Description
Time Sheet Ref	AutoNumber	Primary Key
Staff No	Text	Foreign Key (to Staff table)
Week No	Number	

5. Create a table with the following fields and save it as **Time Sheet Lines**:

Field Name	Type	Description
Time Sheet Ref	Number	Primary Key
Line Ref	AutoNumber	Primary Key
Project Code	Text	Foreign Key (to Project table)
Time Worked	Number	

6. The **Primary Key** for this table is **Time Sheet Ref** and **Line Ref**. Highlight both lines in the table design window before clicking the **Primary Key** button. **Time Sheet Ref** is also a **Foreign Key** linking back to the **Time Sheet Header** table.

7. Make sure all tables are saved and closed, then create the following relationships between all the tables. Ensure that **Referential Integrity** and both **Cascade** options are specified for each link.



8. Close the **Time Sheet** database, saving any unsaved objects.

Note: *The required data records for the tables will be added in a later exercise.*

Summary: Creating and Linking Tables

In this Skill Set you have seen how to design and create new databases and tables. You will have seen how to create a variety of relationships between tables in a database and should be able to understand how relationship properties such as referential integrity and cascade options can affect overall data integrity.

You have also learnt some ways in which data integrity may be maintained using primary keys and other indexes and seen how a relational database can be more efficient than a single table database.

Your OCR ITQ evidence must demonstrate your ability to:

- Design databases:
  - Use of indexes and keys to organise data
  - How relationships are established
- Understand data integrity
  - Primary Keys
  - Methods of maintaining data integrity in a multi-table database
  - Referential Integrity
  - Foreign Keys
- Understand relationships between database tables
  - One to one
  - One to many
  - Many to many
- Define field characteristics
  - Data types