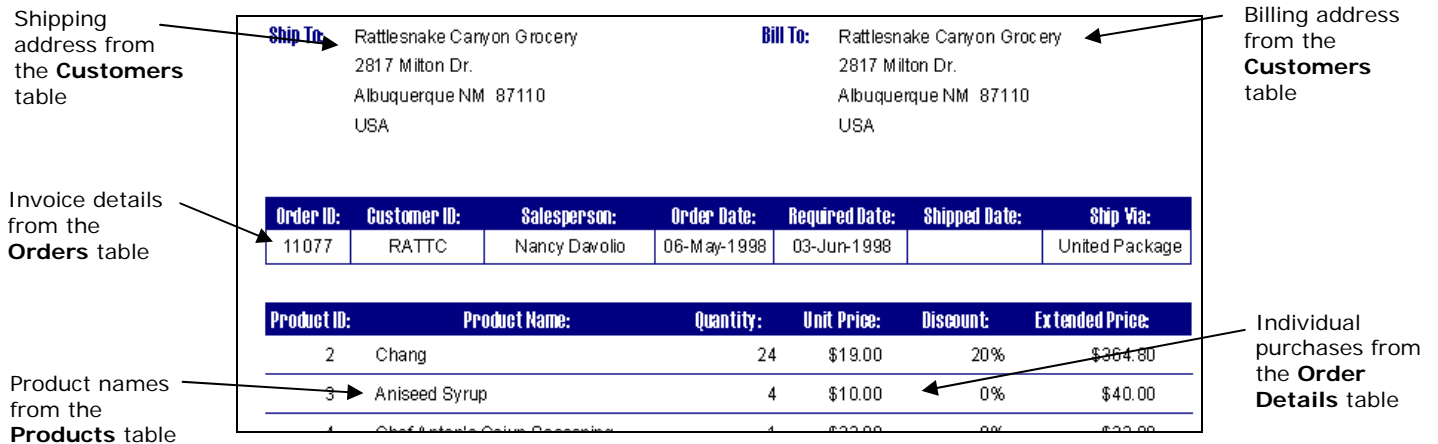


A relational database is powerful because it allows you to gather information together in a query, form or report. For example, the report below is taken from the Northwind sample database. It is compiled by taking information from four different tables.



Shipping address from the **Customers** table

Invoice details from the **Orders** table

Product names from the **Products** table

Billing address from the **Customers** table

Individual purchases from the **Order Details** table

Ship To:	Rattlesnake Canyon Grocery 2817 Milton Dr. Albuquerque NM 87110 USA	Bill To:	Rattlesnake Canyon Grocery 2817 Milton Dr. Albuquerque NM 87110 USA			
Order ID:	Customer ID:	Salesperson:	Order Date:	Required Date:	Shipped Date:	Ship Via:
11077	RATTC	Nancy Davolio	06-May-1998	03-Jun-1998		United Package
Product ID:	Product Name:	Quantity:	Unit Price:	Discount:	Extended Price:	
2	Chang	24	\$19.00	20%	\$364.80	
3	Aniseed Syrup	4	\$10.00	0%	\$40.00	
4	Chef Antoinette's Cajun Rice	4	\$22.00	0%	\$88.00	

Before information from numerous tables can be collated like this, it is necessary to link the tables together through relationships. We have previously discussed how identical fields play a part in a relationship. We now need to look at the other aspects of creating our relational database.

Types of Relationship

When creating a relationship between two tables, Access has to understand what type of relationship it is going to be. The three possibilities are as follows:

1. One-to-Many relationships
2. Many-to-Many relationships
3. One-to-One relationships

One-to-Many Relationships

These are used when one record in the first table matches a number of records in the second table. For example, each customer in the 'Customers' table may have a number of orders in the 'Orders' table. This is a 'one-to-many' relationship.

Many-to-Many Relationships

These are used when each record in the first table can match a number of records in the second table and each record in the second table can match a number of records in the first. Imagine our company bought a single product from a number of different suppliers. Each record in the 'Suppliers' table could match a number of different products in the 'Products' table, but each product in the 'Products' table could also match a number of suppliers. This is a 'many-to-many' relationship.

Note – 'many-to-many' relationships can cause problems. In the example above, you would have repeated information about the same product in the 'Products' table. This is inefficient and leads to errors. Database designers will usually find ways to solve the problem, using an extra table and two 'one-to-many' relationships. Think through how you could solve the problem above.

One-to-One Relationships

These are rarely needed. They are used when each record in the first table matches no more than one record in the second table, and each record in the second table matches no more than one record in the first. For example, you may want to send Christmas cards to the top 10% of your customers. As the Christmas card list is a bit of a one-off, you may decide to keep this information in a separate table. This also means that you don't have blank fields for all the customers who aren't going to receive a card. In this case, each customer in the 'Christmas Card' table would match the same customer in the 'Customers' table. This is a 'one-to-one' relationship.

Task 1 – Types of Relationship

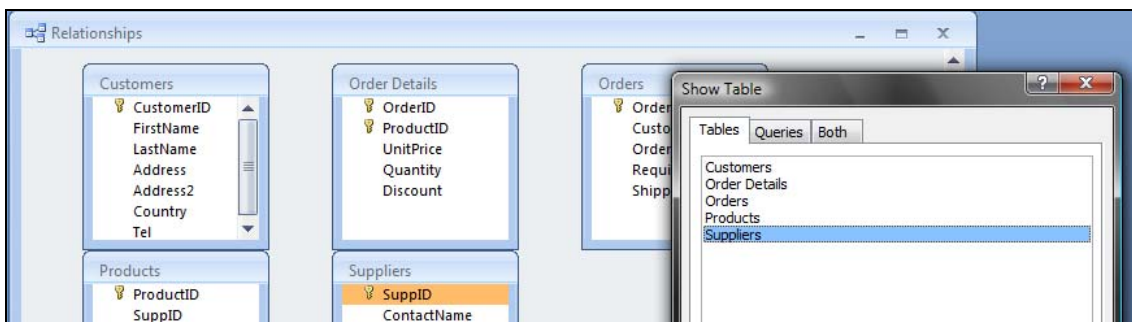
Decide whether each of the following relationships would be 'One-to-Many', 'Many-to-Many' or 'One-to-One'.

- Orders from the 'Orders' table relating to the individual purchases in the 'Order Details' table.
- Products from the 'Products' table relating to the products in the 'Order Details' table.
- Suppliers from the 'Suppliers' table relating to the suppliers in the 'Products' table.

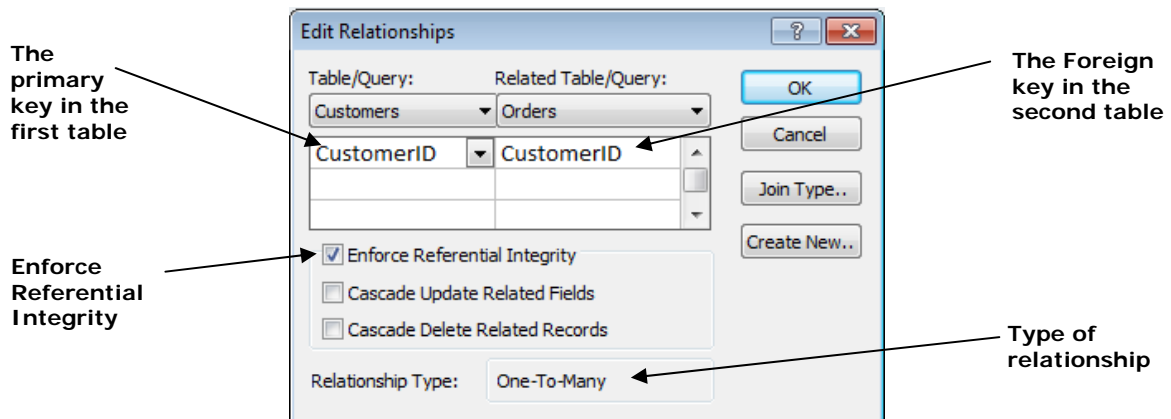
Task 2 – Creating a Relationship

To set up the relationships, we use a schema. A schema is a diagrammatic representation of our database.

- Open your 'Relational' database containing the five tables created earlier.
- Open the 'Database Tools' tab and click on 'Relationships'. Now click on the 'Show Table' button.
- Double-click on each of the five tables in the 'Show Table' window. Close the window when all tables have been added to your schema. You can select and delete any tables added twice by mistake.



- Layout the tables in the order that they are to be linked i.e. Customers – Orders - Order Details – Products - Suppliers. You can increase the size of the window if necessary.
- Click on the 'CustomerID' field in the 'Customers' table and drag the cursor over the 'CustomerID' field in the 'Orders' table. When the mouse button is released, Access will open a window displaying the relationship that it presumes you want to create.



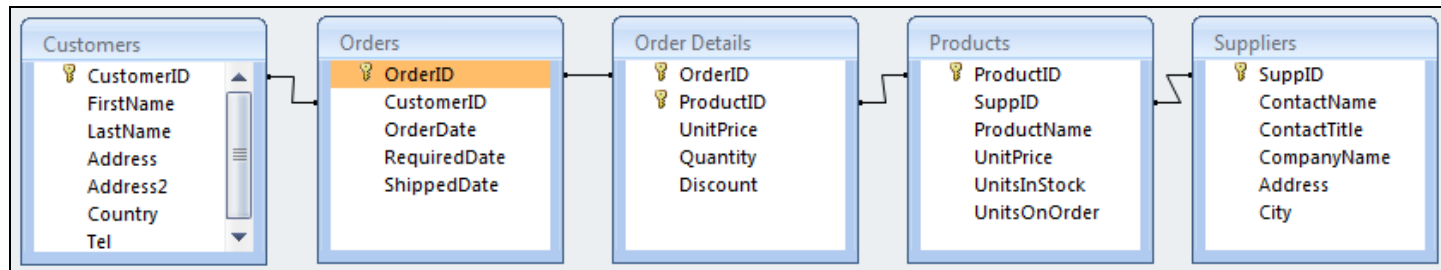
- The window should display the 'CustomerID' field in both tables. If it doesn't, then you have not dragged the field over correctly. The correct fields can be selected from the drop-down lists.
- Click the 'Create' button. A line will appear between the two tables. The relationship can be edited by right-clicking on the line and selecting 'Edit Relationship'. It can be deleted by right-clicking on the line and selecting 'Delete'.

Task 3 – Completing the Relationships

We have created one relationship in our database. It is now necessary to repeat this process for the other relationships. Remember, in each case:

- Drag from the primary key field in one table to the identical field in the other.
- Check that the correct fields are involved.

Note – in this case, all relationships are created by dragging towards the centre in the schema.



Task 4 – Referential Integrity

When creating relationships between tables, it is possible to set referential integrity. Referential integrity is essentially a set of rules that the database uses to make sure that a relationship is maintained. For example, each of the orders placed in our 'Orders' table comes from a customer. Referential integrity will stop an order being placed if the customer is not in the 'Customers' table.

Referential integrity can also stop data being deleted accidentally. For example, each product in the 'Products' table will have been supplied by someone in the 'Suppliers' table. Referential integrity will stop a supplier being deleted if they are still linked to products in the 'Products' table.

Referential integrity can only be used under certain conditions. These are as follows:

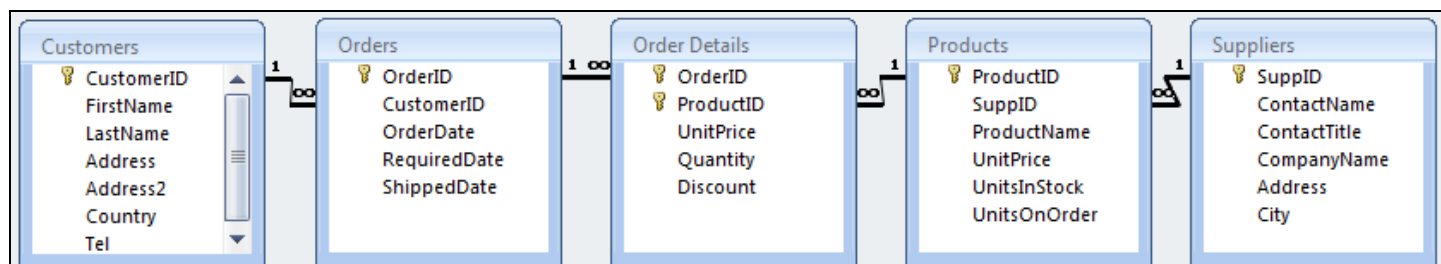
- the field in the first (or primary) table is the primary key in that table.
- the related fields in both tables have the same data type.

When the 'Enforce Referential Integrity' option is selected for a relationship, you are limited in the way that data can be entered. For example:

- a supplier must be entered before the products they supply.
- a customer must be entered before an order can be placed by them.
- an order must be created before the details can be recorded in the 'Order Details' table.
- all products provided by a particular supplier must be edited or deleted before removing the supplier.
- the primary key in the first table cannot be changed whilst related records exist in the second table.

Note – Access provides a 'Cascade' facility to override these limitations and still preserve referential integrity. For example, if you change a 'CustomerID' in the 'Customers' table, it will change the 'CustomerID' for that customer in each related record in the 'Orders' table. However, because these changes can't easily be reversed, it is suggested that this facility is generally not used.

Task – edit each of the relationships so that referential integrity is enforced. Symbols above the join line in the 'Relationships' window will appear. These indicate the type of relationship: one-to-one, one-to-many etc. The '1' indicates 'one', while the infinity symbol (∞) indicates 'many'. All our relationships should be one-to-many.



Task 1 – Types of Relationship

Decide whether each of the following relationships would be 'One-to-Many', 'Many-to-Many' or 'One-to-One'.

- a. Orders from the 'Orders' table relating to the individual purchases in the 'Order Details' table.

One-to-Many

- b. Products from the 'Products' table relating to the products in the 'Order Details' table.

One-to-Many

- c. Suppliers from the 'Suppliers' table relating to the suppliers in the 'Products' table.

One-to-Many