Customer

CustID	Name	FirstName	Street	Town	PhoneNo
1	Sykes	Jim	2 High Road	Greenwood	01395 211056
2	Perle	Lee	14 Duke Street	Greenwood	01395 237851
3	Hargreaves	Les	11 Forest Road	Prestwich	01462 501339
4	James	Sheena	4 Duke Street	Greenwood	01395 237663
5	Robins	Charlie	11Juniper Road	Greenwood	01395 267843

Payment

Payment No	CustID	Date	Total amount paid	Total deposit paid	Total deposit returned
401	4	19/03/04	£56.00	£50.00	£50.00
402	20	19/03/04	£20.00	£25.00	£25.00
403	4	19/03/04	£145.00	£80.00	£80.00
404	3	20/03/04	£186.00	£100.00	£84.00
405	2	20/03/04	£44.00	£40.00	£40.00

Figure 9.17 *One to many association between the Customer and Payment classes implemented as two tables with a foreign key*

The second way of implementing a one to many association is to include a foreign key[5] in the table for the many class. This method of implementation has the advantage that it results in fewer tables, which means that navigation around the database is simplified. The two tables implementing the association between Customer and Payment are shown in Figure 9.17. Note the extra field, CustID, in the Payment table – this is the foreign key. In this example it is customer number 4 who is associated with more than one payment.

When converting relationships in a class diagram to tables in a relational database, aggregation is treated in the same way as a one to many association.

Many to many associations. Many to many associations are always implemented as in the first method shown for one to many associations. Separate tables are created for each class and for the association.

[5] *If one table contains the primary key of another, this is called a foreign key. A foreign key permits a link between the two tables.*