

Attributes

The attributes of an entity are the items of data that belong to that entity and that we need to store for the purposes of the database. The attributes of STUDENT in the table on page 2 are Name, Set, Tutor Group, Tutor and Grades.

The attributes of an entity is a list of the data items that we need to store for the entity class. It must include all the data that belongs to the entity and that is also needed for the purposes of the database. If we needed to analyse student performance by gender then sex would be needed as an additional attribute of STUDENT.

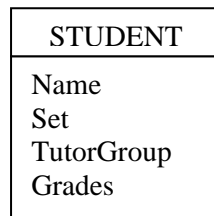
The actual list of attributes for a given entity will depend on the entity itself and the information that the database must provide. It would be quite possible for two different database applications to contain the same entities but to have different attributes associated with them.

If you are familiar with flat-file databases then the entities can be thought of as corresponding to records and their attributes to fields.

There are two commonly used ways of showing the attributes of an entity. The first way is to write the list of attributes in brackets after the entity name. The entity STUDENT could then be written:

STUDENT (Name, Set, TutorGroup, Tutor, Grades)

An alternative method is to show the information in diagram form.

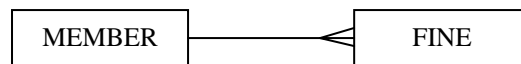


Relationships

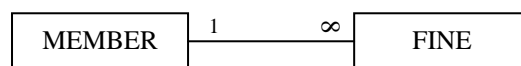
Normally a database will store data about more than one entity. In the library database, for example, we had BOOK, MEMBER, LOAN and FINE. We can identify relationships between these entities. A relationship, if it exists, will be one of three types.

A one-to-many relationship exists when it is possible that one instance of entity A might be linked to many instances of entity B but, when the view is reversed one instance of B links to only one instance of A. This is easier to understand when applied to some actual entities.

One particular member may have paid many fines. One particular fine will have been paid by one particular member. The relationship between MEMBER and FINE is therefore a one-to-many relationship. It would be shown in an entity-relationship diagram in the following way.



The 'crow's foot' lines indicate the many side of the relationship. An alternative method of showing the many side of the relationship is to use the symbol for infinity to indicate the many side of the relationship and a 1 to indicate the one side. In this case the E-R diagram would look like this:



These are just two slightly different ways of showing exactly the same thing – a one-to-many relationship between MEMBER and FINE.

It is important to realise that the relationship indicates what is possible, not what has necessarily happened. The one-to-many relationship between MEMBER and FINE shows that it is possible for one member to have many fines. A given instance of MEMBER may have paid only one fine or

Chapter 4 Tables and First Normal Form

Representing Entities

We are now almost at a stage where our entities can be represented by tables that can be directly implemented in a relational database. The table used to represent an entity should obey the following rules.

- (1) The table must have a unique name. Usually this will be the plural of the corresponding entity, so that the entity *STUDENT* is represented by the table *tblStudents*. The *tbl* is added to the front of the name because, when we implement the table, it is useful to be able to identify tables as opposed to other objects like queries or forms.
- (2) The table must be made up of columns and rows. Each column must have a unique name and the order of the columns is unimportant. The column names will be the attributes that we have identified for the entity
- (3) No two rows can be identical. We have ensured this by selecting one or more attributes to be an entity identifier. In the table we will call this the primary key.
- (4) The order of the rows is unimportant. Each row stores data about one instance of the entity. In a truly relational database the order in which the rows are placed cannot be important.
- (5) Each cell in the table must contain a single indivisible data value. We have not addressed this requirement yet. It will mean that we have to remove lists of data – such as the list of days that a doctor attends a particular surgery – from our entities.

The set of rules above come from mathematics and defines a mathematical concept called a relation. The correct name for our table is in fact relation. We say that we implement the entities in a relation. It is this that gives the relational database its name – because we use relations to represent the entities. You probably thought up to now (quite naturally) that we called the database relational because of the relationships between the entities. But it is the mathematical relation, with its rows and columns and strict mathematical definition that gives this type of database its name.

Tables, Access and Relational Databases

As we start to represent the entities of the data model in the form of relations or tables, our design begins to match up with what will happen at implementation. Each entity will be represented as a table in the database. The attributes will be represented as columns and the combination of attributes that define an instance of the entity will be represented by a row.

Figure 4.1 shows the pupil data from the table on page 2 converted to an Access table. This table does not comply with the definition of a relation. There is no guarantee that each row will be unique which breaks rule (3) above. Also the values in the Grades column can be broken down into a number of separate values, which breaks rule (5).

Chapter 6 Third Normal Form

Third Normal Form

Third Normal Form A table is in third normal form if it is in second normal form and there is no functional dependency between non-key items.

Third normal form requires that each of the informational or non-key item columns of the table is independent of any other.

There are two ways in which one informational item can be dependent on another. Firstly if one column value is calculated from another then there is obviously a functional dependency. Secondly we may have a situation where there is a non-calculated dependency between two items.

Calculated Dependency

Suppose we had an entity CUSTOMER (AccountNo, Name, Address, DOB, Age). A table representing this entity would be in 2NF since Name, Address, Date of Birth and Age are functionally dependent on AccountNo. (Assuming that Address is atomic for the purposes of the application).

Age can be calculated from DOB and there is therefore a calculated dependency between them. A calculated dependency is removed by removing the calculated value. It is not usually necessary to store a calculated value since it can be calculated from the underlying data as and when required.

There are some situations where we would want to store a calculated value. We may need to keep the original calculated value when the underlying value changes – perhaps for audit purposes. Another case where we might want to store calculated values is when a large number of calculations have to be done in order to display information. This might degrade database performance and so we might decide that it is better to store the calculated values in defiance of 3NF.

This raises an important issue. Normalisation is a guide to developing a database structure that results in minimum data duplication and maximum data independence. There is no point in achieving these goals if the resulting structure does not fulfil the needs of the end user. There will be situations where the best design strategy will be to have tables that are not fully normalised. Database design is therefore not an exact mathematical process – it relies on the designer's experience to know when to break the rules.

However, normalisation is the safe approach and it should only be discarded to achieve clearly defined objectives and only once the benefits have been weighed against the disadvantages of using non-normalised data structures.

Practical 09.01

Follow the steps outlined below to implement *frmEditPupilData*. You should have a copy of the design plan for the form (which you will find in file 09P01.doc) in front of you when you do this.

Use the copy of your database that you made before you started the testing in practical 08.02 alternatively may use the copy of the database provided as file db09P01.mdb. This database contains the tables with referential integrity, cascade updates and cascade deletes set up. It also contains the data from data set 1 used at the end of the last chapter but without any of the changes to the data made as part of the testing for practical 08.2.

As you create the form, keep track of what you have done by ticking each element on the design diagram as you implement it.

	<p>Remember that you should have a copy of the design plan (the Word document 09P01.doc) while you are working through this section.</p> <p>Select the <i>Forms</i> object in database view and click <i>New</i> to create a new form.</p> <p>When the new form dialog appears make sure that you have <i>Design View</i> selected. You will also need to tell Access which table or query the form is to be based on.</p> <p>Select <i>tblPupils</i> from the drop down list and then click <i>OK</i>.</p>
	<p>Make sure that you are familiar with the various elements of the design screen for a form. The picture on the left shows the blank form, the toolbox, the field list and the properties window.</p> <p>If the toolbox, field list or properties window is not visible then they can be opened or closed either by using the view menu or by using the appropriate tool on the toolbar.</p> <p>Field List</p> <p>Toolbox</p> <p>Properties Window</p>
	<p>The form is divided up into several different sections. The properties window will show the properties of the section that you currently have selected.</p> <p>Click anywhere on the Detail Section (the blank form) and you will see the properties window change from <i>Form</i> to <i>Section: Detail</i>.</p> <p>Get back to the Form properties by using <i>Edit/Select Form</i>. With the Form properties showing, change the <i>Caption</i> Property to <i>Edit Pupil Data</i>.</p> <p>Tick your design sheet to show that you have implemented the caption design. (Top right hand corner of the sheet under <i>Other Information</i>.)</p>