Higher Still
Notes

www.hsn.uk.net

Higher Information Systems

HSN31010
Database Systems
First Normal Form

Contents

Normalising to First Normal Form

What is Normalising?

Normalising is a process carried out on the entities of a database in order to make the database system more efficient. In particular, normalising will help with setting up a relational database (a part of the Higher coursework), so it is important to know what to do. As well as this, there is a question on normalising in the exam, worth 10 marks.

Normal Forms

Normalising is a very logical process, and there are several stages which we must go through. Each stage improves the efficiency of the database system. The stages are called 'Normal Forms', and for the purposes of the Higher course, there are three you need to be aware of:

- First Normal Form (1NF)
  To achieve 1NF, we must remove 'repeating groups' of data.
- Second Normal Form (2NF)
  To achieve 2NF, partial dependencies on the primary key must be removed.
- Third Normal Form (3NF)
  To achieve 3NF, dependencies on non-key fields must be removed.

In most cases, you will only be expected to produce a model in 1NF (although it may be necessary to go further to achieve the best solution to the coursework). In the exam, you should only be asked to work to 1NF, but there may be questions requiring some knowledge of 2NF or 3NF.

The Process

The process for normalising to 1NF is generally straightforward, and can be tackled in much the same way for every question. The process for producing the 1NF data model is as follows:

1. Identify all the data items
2. Identify any repeating groups
3. Identify the key(s)
4. Remove any repeating groups into separate entities
5. Identify keys, add foreign key(s) to represent the relationships

Most exams will roughly require this procedure to be followed, although you may be guided through the process.

Worked Example

To illustrate the application of the process for 1NF, we will work through the following example, in which a company keeping paper records of staff in its shops wishes to computerise the database. Some sample data is shown below:

Shop ID	Shop Location	Telephone No.	Staff ID	Staff Name	Post
7262	Edinburgh	0131 4349816	5242AB	Karen Wilson	Manager
			7262DG	Harry Jones	Supervisor
9928	Glasgow	0141 5726481	9882UY	Gina Ross	Manager
			4433QW	Lesley Pugh	Supervisor
			6523GC	Fred Kinder	Sales Assistant

The company has also explained that a member of staff only ever works in one shop at a time; each Staff ID is unique to a member of staff and each Shop ID is unique to a shop.

1. Identify all the data items

This will also involve identifying the primary entities. In this simple example, there is only one entity, with data items as follows (reading the headings of the table):

Staff (Shop ID
Shop Location
Telephone No.
Staff ID
Staff Name
Post)

2. Identify any repeating groups

The repeating group is the set of data items "Staff ID", "Staff Name" and "Post", since these have more than one value in each instance. This can be shown on the primary entity:

Staff (Shop ID
Shop Location
Telephone No.
(Staff ID
Staff Name
Post))

3. Identify the key(s)

Because there is only one entity, there can be no foreign keys. However, we must establish the primary key of the entity. The primary key is the field(s) which are unique in every instance. Looking at the sample data, it would seem that Staff Name could be the primary key – there are no two staff with the same name. However, this would mean no new member of staff could have the same name as an existing member of staff.

We are in fact told that Staff ID is a unique code for each person, so it follows that it will be unique in every instance of the entity. Staff ID is the primary key:

Staff (Shop ID
Shop Location
Telephone No.
(Staff ID
Staff Name
Post))

We have now modelled the un-normalised system. The next step will begin the normalisation.

4. Remove any repeating groups into separate entities

The repeating group in Staff can be removed to a new entity, which we will call Staff:

Staff (Staff ID
Staff Name
Post)

The existing entity is renamed Shop:

Shop (Shop ID
Shop Location
Telephone No.)

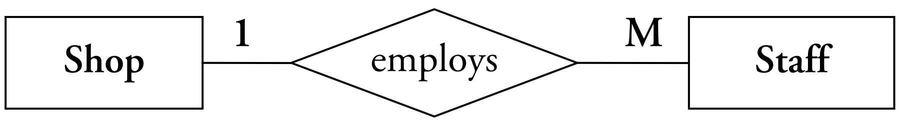
5. Identify keys, add foreign key(s) to represent the relationships

We know from the original data that one shop has many staff; a one-to many relationship between shop and staff. The foreign key is always placed at the 'many' end of the relationship, so we have to add a foreign key to Staff. The data item we add comes from the primary key of Shop, Shop ID.

Shop (Shop ID
Shop Location
Telephone No.)

Staff (* Shop ID
Staff ID
Staff Name
Post)

In addition, you will usually be asked to draw an entity relationship (ER) diagram. In these diagrams, a box represents an entity and a diamond describes the relationship between them:



The **1** and **M** show the nature of the relationship between the entities.

Why do we Normalise?

It might seem that there is no real benefit in normalising entities, since we still have the same data items, only now we have more entities. While it might seem confusing, it actually makes the database much simpler to use on the computer. There are three general problems with un-normalised data models:

1. Adding Records

   If we want to add a new shop which has not yet appointed staff, we would have to insert a record leaving fields blank. In the un-normalised model, this would involve omitting the primary key, which we cannot do.

2. Deleting Records

   If all the members of staff in a particular shop decide to leave, their entries would be deleted. This would also delete the only record of the shop's location and telephone number, which we do not want to do.

3. Modifying Records

   If a shop changes its telephone number, we would have to change every instance containing the old number. In the un-normalised system, this would amount to the number of staff employed in that shop. When retyping so many values, it is clear mistakes could be made, and perhaps not all the instances will be corrected. This would compromise the integrity of the data.