Specification Reference Section 14.3 – Database Management Concepts
Heathcote 2nd Edition – Chapter 55-57, not included in 1st Edition.
Mott and Leeming Chapter 25

What is a database?

A database is simply a collection of data? These days the term 'database' usually refers to a collection of data on a computer but it is quite possible to have non-computerised data bases e.g. a card index.

Types of Database

The types of database refer to the way in which the data is stored. The types we will consider are:

(a) Flat files – really to show their disadvantages compared to more sophisticated systems.
(b) Relational databases – we will look in some detail at these.

Flat File System

A flat file is a database held in a single file. A flat file database can be implemented using a spreadsheet package treating each row as a record and the columns defining the fields. You have probably already done this for simple applications. Basic manipulation of the data is possible e.g. sorting, searching etc. but in order to provide more sophisticated and flexible processing a Database Management System (DBMS) is required. Microsoft Access and FileMaker Pro are examples of DBMS.

Disadvantages of Flat File Systems

Suppose we wanted to store the teaching arrangements for a school in a flat file system. We might decide to treat every student's subject allocation as a separate record. An extract from the files could look like this.

Surname	**Christian Name**	**House**	**Year**	**Subject**	**Group**	**Teacher**	**Room**
Jones	John	Lancaster	5	English	X	MAR	25
Smith	Mary	Beaufort	4	Maths	A	GAM	13
Walker	Tom	Salisbury	5	English	X	MAR	25
Jones	John	School	5	Maths	A	GAM	7

It should be immediately obvious that there are a number of undesirable features:

Redundancy of data – Every record for a particular student contains several items of data which are repeated. This wastes space/memory and leads to other problems – see below.
Data is error prone – Because data is repeated, it is easy to introduce errors and inconsistencies. In the above example John Jones is entered as being in Lancaster House in one record and School House in another. There is no way of knowing which is right. Could they both be right?

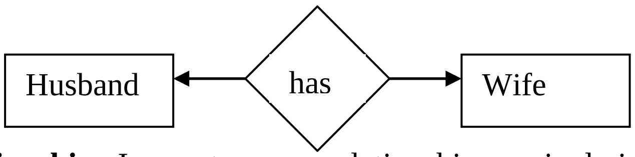
Difficult to update or modify the data – What would we need to do if the teacher for Group X English was changed?

Multiple files holding same data – It is quite likely that other areas of the school would have their own systems holding some of the same data. For example the library could have a system holding the same information about pupils. In a large organization there could well be many systems all holding what should be the same data but probably is not.

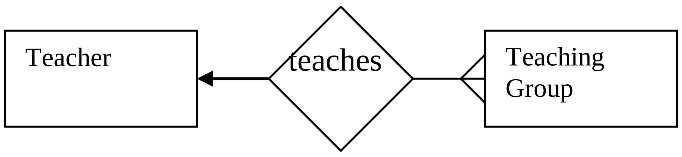
Data Modelling
In order to overcome these disadvantages, we need to take a more intelligent approach to storing the data and have some software which will provide us with useful data manipulation facilities. This software is the DBMS. The more intelligent approach to storing the data is known as data modelling.

Instead of throwing all the data which is relevant to a particular problem into one table, we need to consider real world ‘things’ which are involved and define the relationships between them. These ‘things’ are called entities and may be actual physical entities (e.g. student) or logical concepts (we will see examples of this later). If we consider the data in the flat file system we have already discussed, what entities can you define?

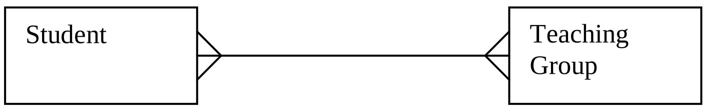
Entity Relationships
There are a number of possible relationships types.
One to one relationships In a one to one relationship every instance of Entity A is related to one and only one instance of Entity B. For example, if we consider the entities ‘husband’ and ‘wife’, there is a one to one relationship because every husband can have only one wife and vice versa.



One to many relationships In one to many relationships, a single instance of entity A can be related to multiple instances of Entity B. For example, a teaching group can have only one teacher but a single teacher can teach many teaching groups.



Many to many relationships In many to many relationships, a multiple instances of entity A can be related to multiple instances of Entity B. For example, a student can belong to many teaching groups and a teaching group contains many students.

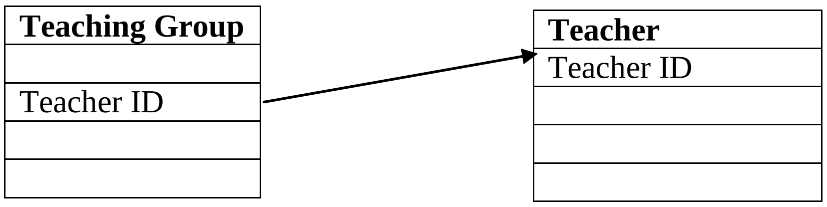


Define suitable entities and draw entity relationship diagrams for the following:
1. A library loans books to its members.
2. A medical practice has a number of doctors. Patients are registered with a particular doctor of the practice. Doctors issue prescriptions to patients for drugs.
3. At a horse-racing meeting, there is a trophy for every race. Every horse has an owner and a jockey. A horse cannot run in more than one race.

Attributes
Having identified relevant entities, we need to identify the information we need to keep about each. This information is stored in the form of attributes i.e. single data items which apply to the entity.

Hence, the entity STUDENT may have the attributes Forename, Surname, Age, DOB etc. etc.

In general, relationships are defined by linking attributes. In the Teaching Group to Teacher relationship above, an attribute of Teaching group would be Teacher and the value of that attribute for a given instance would be a unique identifier of the Teacher. In database terms, this would be the value of the primary key in the Teacher table.



We can now see the idea of a dependency. The concept of Teaching Group depends on their being a Teacher for it to make sense. The arrows in the ER diagrams show the direction of the dependencies involved. Note for the Husband/Wife relationship, they are mutually dependent i.e. it is not possible to have a husband without a corresponding wife, and vice versa.

Standard Notation for Defining Database Tables
There is a standard notation for writing down the definition of the tables for a database. Consider the following for a STUDENT:

STUDENT (<u>Student Number</u>, Surname, Forename, Age, House, $\overline{\textit{Subject Studied}}$, *Tutor*)

The name of the entity/table is in upper case at the beginning of the statement and the fields are within the parentheses separated by commas. Note that the table name is singular. There are a number of conventions for identifying the fields:

The primary key is underlined.
Foreign keys are in italics. A foreign key is the primary key of another table and defines the relationships between the tables. Hence, we would expect there to be a TUTOR table within the database.
Repeated fields have a line drawn above them. Hence, a student studies multiple subjects and we would expect there to be a SUBJECT table in the database. We will soon see that we get rid of repeated fields in real applications but we may well have them at the analysis stage.

Database Normalisation
The process of normalization of a database design means making the design as efficient as possible, which effectively means to minimize data redundancy. Although there are a number of rules, we will only concern ourselves with the first three. A database which conforms to rules up to rule N is said to be in Nth Normal Form. Hence, we will be aiming at 3rd Normal Form designs.

GBH Photo Suppliers
Let us suppose that this imaginary firm processes orders, a typical example of which is shown below:

GBH Photo Suppliers

Order No.	7568		**Date** 13 Jan 2002
Account Number	23768		
Customer	Tom Jones		
Address	47, High St.		

Item Code	**Description**	**Quantity**	**Unit Cost (£)**
4051	Olympus 700XB	1	69.99
1743	Pack of Film	4	9.98
2371	Single Use camera	3	7.49

Total Order Cost = £132.38

We will start from a situation where orders are stored in a single flat file. Write down a definition of the table ORDER which would include all the essential information for an order.

First Normal Form

A table is in first normal form if it has no repeating attributes or groups of attributes.

Second Normal Form

A table is in second normal form if it is in first normal form and no attribute that is not part of the primary key is dependent on only a portion of the primary key.

Third Normal Form

A table in third normal form contains no ‘non-key’ dependencies.

Zero Normal Form ORDER Table

ORDER (Order Number, Order Date, Account Number, Customer Name, Customer Address, Item Code, Item Description, Item Quantity, Item Unit Cost, Total Order Cost)

We can remove the repeating fields and form a new table called ITEM ORDER and relate this table to the ORDER table. Hence, a record in this table will refer to a particular item on a particular order. The primary key, therefore is a combination of Order Number and Item Code because both are required to define the record uniquely. The tables now look like:

First Normal Form Tables

ORDER (Order Number, Order Date, Account Number, Customer Name, Customer Address, Total Order Cost)

ITEM ORDER (*Order Number*, Item Code, Item Description, Item Quantity, Item Unit Cost)

If we examine the ITEM ORDER table definition it is clear that some fields depend on Item Code but are independent of Order Number. These are Item Description and Item Unit Cost. Hence, these fields are dependent on only part of the primary key and therefore contravene the rule for second normal form. The solution is to remove them to a new table called ITEM.

Second Normal Form Tables

ORDER (Order Number, Order Date, Account Number, Customer Name, Customer Address, Total Order Cost)

ITEM ORDER (*Order Number*, *Item Code*, Item Quantity)

ITEM (Item Code, Item Description, Item Unit Cost

Examining the ORDER table, the fields Customer name and Customer Address are related to the key field Order Number only through the field which uniquely defines the customer i.e. Account Number. There is no direct relationship between Order Number and Customer name or Customer Address; hence, these fields contravene the rule for third normal form. Again the solution is to remove them to a new table called CUSTOMER:

Third Normal Form Tables

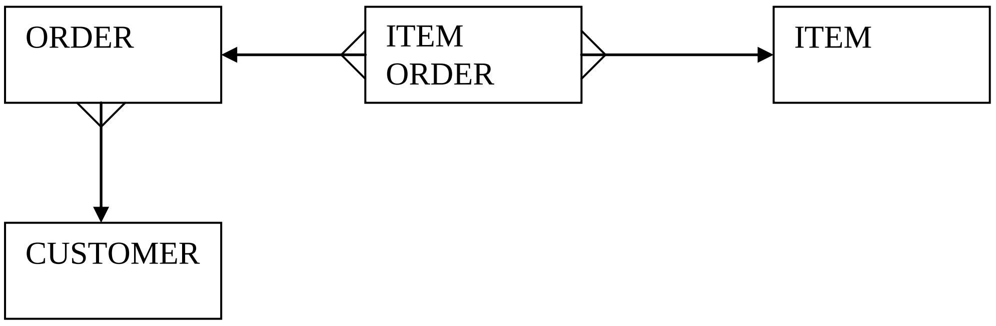
ORDER (Order Number, Order Date, *Account Number*, Total Order Cost)

CUSTOMER (Account Number, Customer Name, Customer Address)

ITEM ORDER (*Order Number*, *Item Code*, Item Quantity)

ITEM (Item Code, Item Description, Item Unit Cost)

The entity relationship diagram for these tables is:



The records in the tables for this order would be:

ORDER(7568, 13/1/2002, 23768,£132.38)

CUSTOMER(23768, Tom Jones, 47 High Street)

There would be three relevant ITEM ORDER records i.e.

ITEM ORDER(7568, 4051,1)
ITEM ORDER(7568, 1743,4)
ITEM ORDER(7568, 2371,3)

There would be three relevant ITEM records i.e.
ITEM(4051, Olympus 700XB, 69.99)
ITEM(1743,Pack of Film,9.98)
ITEM(2371, Single Use Camera,7.49)

Homework

(a) Explain what is meant by a flat-file database.
(b) Give an example of an application for which a flat file database would be suitable, giving reasons for your answer.
(c) A telephone ordering firm offers a range of several thousand items which are sold to the public exclusively over the telephone. It has a relatively small number of preferred suppliers. It employs a marketing strategy team and the firm's accountants keep very tight control on all financial aspects of the firm. The firm is to install a new, comprehensive data base system. Suggest facilities that would be required by:
    (i) the telephone answering staff
    (ii) the marketing strategy team
    (iii) the accountants.
    Why would a flat-file system be unlikely to be satisfactory?

(d) A school maintains its pupil records on a spreadsheet with each pupil having a single record (row). An example of the data stored is shown below:

Name: Harry Windsor
Roll Number: 85294
Age: 17
House: Buckingham
House Telephone Number: 496
House Master: JKS

Subjects Studied:	**Subject**	**Expected grade**	**Group**	**Teacher**	**Room**
	Physics	B	PHY3	GBH	PL2
	Chemistry	D	CH1	JFC	C2
	Maths	C	M1	JLD	M5

(i) The table is called STUDENT. Write down a definition of this table using standard database notation.
(ii) Explaining your reasoning, derive a set of tables for this data in third normal form.
(iii) Draw the entity relationship diagram for this data.
(iv) Populate the tables for data relevant to Harry.

.

Database Management

DataBase Management System (DBMS)
This is the suite of software which is used to control and use the database. Examples of facilities it will provide are: File management, data entry, data interrogation giving different ‘views’ of the data, reports, backup, access control and security, user friendly front end (GUI) etc. A useful illustration is provided in Bradley Page 564.

Data Description (Definition) Language (DDL)
This is the language used to define the structure and content of the files in a database. It will define the fields, properties of fields, validations, key fields etc. In PC based databases such as Access or FileMaker, this language is hidden by more user-friendly graphical interfaces.

Data Manipulation Language (DML)
This is the language used to allow modification of the data within a database e.g. creating, modifying or deleting data. Some facilities provided by the DML are only for use by the DBA (see below) but others provide facilities for use by the user e.g. generating queries. Queries would usually be written in SQL Structured (Standard) Query Language which can be considered as part of the DML.

Data Dictionary
This can be thought of as a database about the database. It defines the structure and content of the database including details of tables, fields (name, type, length, allowed values etc.), relationships, access privileges for programs and users etc. It may also contain explanatory comments and descriptions.

Data base Administrator (DBA)
The DBA is the person responsible for designing and maintaining the database. His/her duties would include: designing data structures, carrying out changes requested by users, maintaining the integrity of the data base by suitable backup procedures, controlling access by allocating suitable privilege levels, maintaining passwords, maintaining the data dictionary, monitoring performance (e.g. response times)

Database Access
In general, a database will be accessed by many users. This gives rise to a number of considerations including:
(a) Different users will legitimately need access to different items of data.
(b) Users will require different levels of access to the same data items e.g. read only, read and write (modify), read/write/delete.

A number of security levels will be defined giving different levels of access. Users will be allocated an appropriate security level. A user gains access by logging on with user id and password and this will define the level of access privilege available to the user.
If the database allows multiple simultaneous access, then a locking system must be implemented. Hence, if User A accesses an record in read/write mode, that record will be locked and no other user will be able to access it until it is released by User A. Another user may be given the option of viewing the record in read only mode. Exactly how much data is locked depends on the circumstances and the file organization.

Database Security
Many organizations have become almost totally dependent on their database systems. Hence, the worst thing that could happen is for the system to fail or the data to be lost. Comprehensive facilities for ensuring the integrity and availability of data must therefore be in place. These may include: multiple

copies of data on different physical devices, backups, checkpointing, transactions logs etc. Along with these methods, a comprehensive procedure for recovery from failures will need to be defined.

Three Level Architecture of a DBMS
The technical term for the definition of a database is the schema. The schema is defined at three levels:

(1) The application level which describes the database as seen by the applications which are to use it. There will be many such 'views'.
(2) The logical level which defines the detailed structure of the data in the database.
(3) The physical level which defines how the data is physical organized for storage on the physical storage devices.