Database concepts (1):

Databases and Database management systems (DBMS)

by Robin Beaumont

e-mail: robin@organplayers.co.uk

Last update: Thursday, 21 July 2011

Contents

'Ways of thinking about the clinical information you collect (i.e. Data)' should be read before this chapter.

1. Learning outcomes check list

This chapter aims to provide you with the skills along with useful knowledge which are listed below. After you have completed the chapter you should come back to these points ticking off those with which you feel happy.

Learning outcome	Tick box
Explain what a database is	❑
Describe what a table and field are	❑
Understand the importance of structured data	❑
Describe the standard functions of a DBMS	❑
Explain what meta-data is	❑
Describe a CRUD matrix	❑
Be able to draw a CRUD matrix	❑
Explain what a view is and its relationship to the underlying data	❑
Explain what a primary key is	❑
Explain the difference between a simple and compound key	❑
Explain what an 'oid' is	❑

2. Introduction

The term 'database' is used by everyone now-a-days yet what exactly does the word mean? Most people, not being computer boffins, would find it difficult to provide an explanation but would, if pressed, suggest one or more of the following examples. For example, the method the bank uses to store details of their account, the medical records department of a large hospital or a card index of their own home video collection. The question is what exactly do all these examples have in common?

A good definition of what a database is can be found on the Internet:

> *"A database is one or more large structured tables of persistent data, usually associated with software to update and query the data. A simple database might be a single table containing many records, each of which contains the same set of fields where each field is a certain fixed width."*
>
> *[FOLDOC - Free on-line dictionary of computing Http://wombat.doc.ic.ac.uk/ - adapted]*

This definition clearly needs a lot of explaining, for one thing what is meant by a table? The following sections will take each of the terms in the above definition and provide examples, exercises and explanations.

3. Tables Records and Fields

A database consists of tables which are composed of records which, in turn, are made up of fields.

Key Concept - Database (First definition):

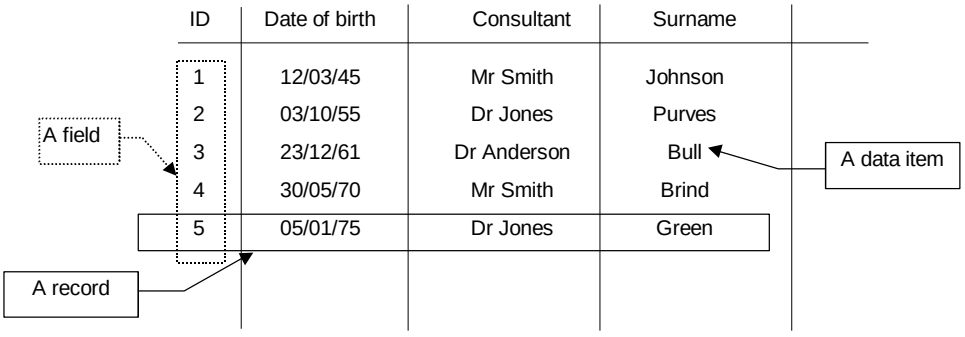
A database consists of

Tables, which consist of records which, consist of fields.

The diagram below provides an example of a typical table consisting of patient details:

A table containing five records each containing four fields

Table name = Patient



In the above diagram:

A row = a **record**

A column = a **field**

A cell = a **data item**; a particular value in a field for a particular record.

A table in a database may be empty, that is contain no records or contain many millions of them. Similarly, a single record may consist of one or thousands of fields. Each field in turn contains a single data item. These data items can be categorised in a number of ways, which we discussed when taking about data types.

3.1 Tables and structured data

From the above it can be seen that a table is basically a collection of data with a name. The name is usually equivalent to some object in the world (e.g. patient, resource, procedure etc.). Initially the tables are equivalent to objects, as perceived by the designers , however as the database assumes greater structure through a process carried out by the modeller known as **normalisation** (covered latter in the course) the number of tables often increases dramatically. The number of tables in a database should not be regarded as equivalent to its complexity.

3.2 Keeping the correct fields together

Considering the next level down, that is the fields within a table a logical question is what, if any, are the rules that result in defining a particular set of fields for a given table. After all the above set of fields in the 'patient details' table are surely largely a matter of personal preference. A dental patient record structure is obviously going to be very different from that of a gynaecologists. The answer to this is rather complex, at the present time it is safe to say that both science and art play a part.

The actual object that the table, as a whole represents (i.e. what its name stands for such as 'doctor'), plays some part in determining the fields, these fields must either be subjected to a formal process known as **normalisation** or use some type of modelling approach (such as UML class modelling techniques) to ensure they should actually be included in the table.

Normalisation is an extremely important process ensuring that a database can actually work. Anyone can come up with all sorts of tables and wonderful sets of fields within each of them but this is paramount to disaster if the data is not structured which means NORMALISED. UML class modelling usually ensures they are normalised. Relational DBMS's, such as Access97, LibreOffice base mySQL etc, are designed to work with normalised data and if presented with non normalised (i.e. unstructured data) will fail to work correctly requiring a lot of code, written by expensive programmers, and sticky tape to keep going. Access97 does provide a facility that takes non-normalised data and supposedly normalises it for you, unfortunately it does not work very well.

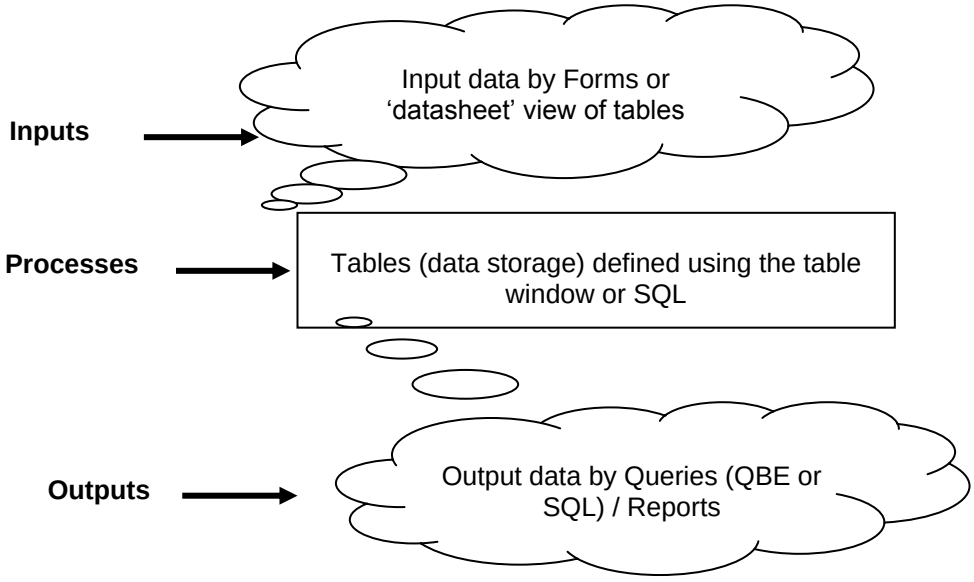
Key points - Designing databases:

1. Object modelling helps structure data into the relevant tables
2. You should always seek advice from a qualified database person when devising databases which have more than a few dozen fields

4. .Database Management Systems (DBMS)

Although databases can be paper based most are computer based. DataBases in computers are usually handled by a special piece of software called a **DataBase Management System** (DBMS). Oracle, Access97 and MySQL are three such DBMSs. Before we discuss the general features of a DBMS we will take a brief look at the database in LibreOffice called base.

Both LibreOffice base and Access97 allows you to do most things in several different ways. For example, you can create the database by usiing the table definition window or you can define it using a standard language called SQL. Similarly you can enter data in one of two ways; the first, **datasheet view** is pretty crude but does the job if you're the only one using the database. Alternatively, you can create pretty screens called **forms** which make the job of entering data easier. You can also manipulate or query your data by using a graphical method (called Query by example **QBE**) or using **SQL** (the grownups database language). You can also create more user friendly versions of these results using **reports** which can provide high quality printed documents.



There is a great deal of information in the above paragraph, and do not worry if you do not understand it, as each of these aspects will be considered in depth latter. The important thing to realise is that a DBMS is much more complex than a spreadsheet or a traditional card indexing system.

In general DBMSs have the following standard features:

1. Meta Data
2. Actions
3. Queries
4. Reporting
5. Views
6. Forms
7. Programming Language

Obviously each supplier constantly tries to outdo the others in offering more features. However we will concentrate on the basic ones listed above.

4.1 Meta-Data

Besides the DBMS storing the actual data you put in it (e.g. names and date of births in the previous patient table example in section 3), it also stores details about this data and the structure in which its kept. That is the DBMS 'knows' how many tables and records there are in the database as well as what data type each field is. All this additional information is called **meta data** and is very useful when you want to find out certain details. The data-dictionary you created earlier is an example of meta-data.

In Access97 you can inspect the meta data that is stored about the current database you have open by clicking on the Tools menu -> Options ->View tab. Then under the Show section, select or clear the Hidden Objects check box and the System Objects check box. Access97 displays hidden objects with dimmed icons to distinguish them from objects that aren't defined as hidden.

4.2 Actions/Privileges

In some respects computer databases (that is DBMS's) work in a very similar way to the old fashion card index boxes loved by most researchers.

Exercise - DBMS Actions

Consider a box collection of files and list the actions you can carry out on them.

A DBMS allows the user to carry out the following activities which are often called user 'rights':

Action:		**Level:**
Creates	-	tables / records / fields
Reads	-	tables / records / fields
Updates	-	tables / records / fields
Deletes	-	tables / records / fields

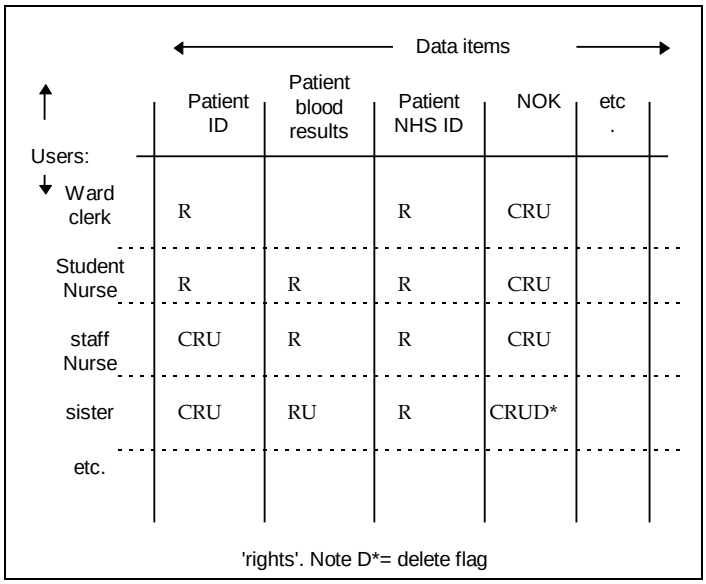
That is the user can possibly be allowed to create, Read, Update or Delete tables, records or fields. These actions / user rights are often referred to as 'CRUD'. The DBMS may also allow the user to carry out ad hoc:

Searches	-	tables / records / fields

The above 'CRUD' is very useful when considering what actions /access rights various people should have to a set of data. For example, a nurse may be able to **R**ead the details of a computerised prescription but not allowed to carry out any of the other actions on the record. Similarly the doctor, who is a colleague of the one who **C**reated the prescription may not be able to **D**elete it. Different professional groups, but notably the administrative managers and the clinicians, will have very different rights to the same sets of data.

Many health systems do not allow anyone to delete anything only what's known as 'delete flag' the data item. This tags the records as being deleted but does not remove it from the database. This is one way of keeping an audit trial, enabling computer user's movements to be recorded. There was recently a case of a GP changing prescriptions, only to be found out by the computers audit trail!

Often database analysts draw what is known as a 'CRUD', matrix. The purpose of a CRUD matrix is to classify who should or does what to various data items (attributes, fields whatever),



'rights'. Note D*= delete flag

CRUD Exercise -

Considering the following three people:

Ward Nurse

Houseman

Unit Manager

Create a 'CRUD' matrix to indicate their type of access ('CRUD'), for the following data items:

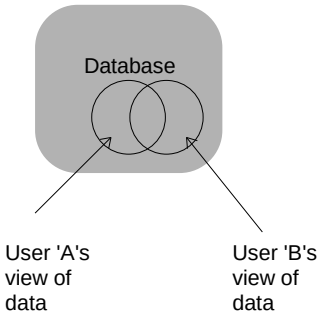
Diagnostic history of patient X	Current bed situation
Current budget deficit	Number of bank nurses on duty
Latest biochemistry result for patient X	Next of kin for patient X

Number of sick days taken by nurses / medical staff

4.3 Queries and reports

The most common complaint concerning databases is the ease of being able to enter data into them compared to the difficulty of getting it back out again. Query and reporting facilities are designed specifically to make the process of extracting data easier.

We will be looking in more detail at queries and less so at reports in the practical sessions to this course.



4.4 Views

From the above it is clear that a database can offer a large number of things to a particular user. In large databases a particular group of users have a particular view of the data. This might well be the result of having few privileges to a very small subset of the database, where as other users will have far greater privileges applied to much more data. The important thing to realise is that what a particular user sees is not necessarily the same as the underlying data. The excellent recent novel 'Quite ugly one morning' by Robert Brookemyre describes this situation in a NHS context.

4.5 Forms

These provide a method of allowing users to enter data in a friendly manner. For example the screen may minic a paper data entry form and provide a method of entering data into several tables simultaneously. A practical session will teach you how to develop them.

4.6 Programming language

Whereas the first DBMS's did not provide a point and click interface, this is now the exception. Such developments have made databases more attractive to a much wider audience with the result that most researchers can now develop a simple Access97 database.

Unfortunately the point and click interface is limited and to develop any database that is usable to any extent must include some programming. Traditionally different aspects of the database had different languages, there was a Data Definition Language (DDL) for creating the database than a Data Manipulation Language (DML) to allow the user to interrogate the database. There were also special reporting languages that provided formatting facilities. Most of these sublanguages have now been seamlessly incorporated into most major programming languages such as Microsoft Visual Basic (the language for Access97).

5. Keys and Indexes

These are both characteristics of tables. The words key and index are often used interchangeably. Traditionally the index was the way the computer implemented the key concept. **A key is a field that has a unique value for each record in the table**, such as a PIN or ID of some sort. There are several variety of keys (superkeys, primary, candidate, foreign etc.) each has slightly different characteristics. At this point, we are only concerned with one type of key:

Primary keys - The primary key is a field or combination of fields that uniquely identifies each record in a table. (e.g. patient ID).

Every table must have a Primary key

Primary Key Exercise 1:

Is the 'ID' field in the table below a suitable primary key. Give reasons for your answer.

A table containing five records each containing four fields

ID	Date of birth	Consultant	Surname
1	12/03/45	Mr Smith	Johnson
2	03/10/55	Dr Jones	Purves
3	23/12/61	Dr Anderson	Bull
4	30/05/70	Mr Smith	Brind
3	05/01/75	Dr Jones	Green

Primary Key Exercise 2:

Is the 'ID' field in the table below a suitable primary key. Give reasons for your answer.

A table containing five records each containing four fields

ID	Date of birth	Consultant	Surname
1	12/03/45	Mr Smith	Johnson
2	03/10/55	Dr Jones	Purves
	23/12/61	Dr Anderson	Bull
4	30/05/70	Mr Smith	Brind
3	05/01/75	Dr Jones	Green

From the above exercise, it is clear that an index field
should not contain an empty ('null') value.

5.1 Types of primary keys

Primary keys can be either simple or composite (compound). A **simple Primary key** is one that is made up of just one field. An example would be if we made the Patient ID the primary key for the Patient table. In this instance, no two patients would ever have the same ID only one record could exist for each patient unless we allowed patients to have more than one ID?

A **composite (compound) primary key** is one where the key consists of more than one field. For example imagine we did not have a unique patient ID field in the patient table. We could possibly say that first name + last name + DOB + post code would give us a unique combination. However this is really a botch for several reasons. Firstly from the previous section I stated that a primary key must possess a valid value. It MUST NOT be empty. Considering our composite key it is highly probable that some of the fields will be empty (null). This means that we can't save any of the other details for the record, such as referral doctor and date placed on list, until we have all the details that make up the primary key. Some patients may be wait a very long time for referrals if this primary key were accepted.

It is always better to have a primary key that is an abstract number rather than one which is made up of several fields which have meaning (see p126 Reingruber & Gregory). In fact the Object Oriented school of modelling considers this so important that they have given primary keys a special name ***oid*** which stands for object ID and which is a characteristic of all objects before anything else is defined.

6. Now check what you have learnt

At the end of every session you are asked to go back to the beginning of the material for the session and read through the 'Learning outcomes check list' again. How many can you tick. If you are not sure about any particular item look up the relevant section if you still have a problem then ask me.

7. Additional Notes

7.1 Primary Keys

Reingruber & Gregory 1994 have the following useful information concerning Primary key characteristics besides the non-null property (see p126). In the extract below where Reingruber & Gregory mention entities consider this as equivalent to objects or tables, the subtleties do not matter at this stage.

A primary key shall be:

1. **Stable.** The value of a primary key must not change or become null throughout the life of an entity (Brooks 1992). A stable primary key helps keep the model stable (Whitener 1989). For example if we consider a patient record the value for the primary key must not change over time as would happen with the age field.

2. **Minimal.** The primary key should be composed of the minimum number of fields that ensures the occurrences are unique.

3. **Factless.** It should not contain intelligence - groupings of digits or characteristics within a value of the key that hold additional meta-information [Note e.g. say patient ID consisted of GP ID + there own unique ID]. This violates atomicity requirements for attributes [note RB: = fields], increases the potential that the primary value would change.

4. **Definitive.** A value must exist for every record at creation time. The primary key acts as a constraining mechanism for the entity. Because an entity occurrence cannot be substantiated unless the primary key value also exists.

5. **Accessible.** Anyone how wants to create, read or delete a record must be able to see the Primary key value (Whitener 1989).

(Adapted from Reingruber & Gregory 1994)

8. References

Reingruber Michael C. Gregory William W 1994 The Data Modeling Handbook John Wiley & Sons. Chichester.

James Martin 1981 An end users guide to Data Base. Prentice Hall [ISBN 0 13 277129 2]

John Carter 1995 The relational database Chapman & Hall