Introduction
Database Technology [DBTECO601]

Thomas D. Devine
http://www.noucamp.org
thomas.devine@lyit.ie

September 8, 2008

Contents

List of Figures

1 Document Information

This document has been produced primarily from the book *Database Systems* [2] and with some material from *Databases Illuminated* [1]. There is no requirement for you to purchase these or any other database book. These notes will be sufficient for any assignments and examination assessment for this module.

2 Introduction

The database [2] is now such an integral part of our day-to-day life that often we are not aware we are using one. To start our discussion of databases, in this section we examine some applications of database systems. For the purposes of this discussion, we can consider a **database** to be a collection of related data and the **Database Management System (DBMS)** to be the software that manages and controls access to the database. We provide accurate definitions later.

When you purchase goods from your local **supermarket**, it is very likely that a database will be accessed. The checkout assistant will run a bar code reader over each of your purchases. This will be linked to a database application program, which uses the bar code to find out the price of the item from a products database. The program then reduces the number of such items in stock and rings the price up on the till. If the reorder level falls below a threshold, the system may automatically place an order to obtain more stocks of that item.

Whenever you visit the LYIT **library**, there is a database containing details of the books in the library, details of the users, reservations and so on. There will be a computerised index, which allows users to find a book based on its title, or its authors or its subject area. The system will also send out reminders to borrowers who have failed to return books on the due date. Typically, the system will have a bar code reader, similar to that used by the supermarket described earlier, which is used to keep track of books coming in and going out of the library.

The LYIT **college** has a database system containing information about you, the course you are enrolled in, details about your grant, the modules you have taken in previous years or are taking this year and details of all your past examination results.

3 Traditional File-Based Systems

File-based system are a collection of application programs that perform services for the end users. Each program defines and manages its own data.

File-based systems were an early attempt to computerise the manual filing system that we are all familiar with. The manual filing system works well while the number of items to be stored is small. It even works quite adequately when there are large numbers of items and we have only to store and retrieve them. However, the manual filing system breaks down when we have to cross-reference or process the information in the files. For example, LYIT

college administration might have a separate file for each student, each lecturer and course. Consider the effort that would be required to answer the following questions:-

- What are the average CAO points for all students enrolled this semester ?
- How many students are from Donegal?
- What is the total annual salary bill for staff?

The manual system is totally inadequate for the information requirements of people working in the LYIT college and many modern day organisations. The file-based system was developed in response to the needs of industry for more efficient data access. However, rather than establish a centralised store for the organisation's operational data, a decentralised approach was taken, where each department, stored and controlled its own data. To understand what this means, let us again consider the LYIT example.

The Administration Department is responsible for the registering and handling administrative queries of students. The file-based system could consist of two files containing student and staff details, as illustrated below. For simplicity, we omit most details.

STUDENT file

student_id	name	address	registered	CAOpoints
s01	Akeroyd	23 Lower Main Street, Letterkenny	2003	350
s02	Thompson	The Maas, Glenties	2007	400
s05	Ellis	Ocean Drive, Dunfanaghy	2007	300

STAFF file

staff_no	name	position
3158	Jennings	Lecturer
3678	Sanderson	Administration
5212	Heathcote	Lecturer
5324	Lai	Maintenance

The Computing Department is responsible for handling the students, lecturing staff and modules taught. The file-based system could consist of three files storing student, staff and module details, containing similar data to that held by the Administration Department, as illustrated below.

STUDENT file

student_id	name	address
s01	Akeroyd	23 Lower Main Street, Letterkenny
s02	Thompson	The Maas, Glenties
s05	Ellis	Ocean Drive, Dunfanaghy

STAFF file

staff_no	name
3158	Jennings
5212	Heathcote

MODULES file

module_code	student_id	lecturer_no
m1	s01	5212
m2	s02	3158
m1	s05	5212

Each department accesses their own files through application programs written specially for them. Each set of departmental application programs handles data entry, file maintenance and the generation of a fixed set of specific reports. What is more important, the physical structure and storage of the data files and records are defined in the application code. We could find similar examples in other departments. For example, the Payroll Department may store details relating to each employee's salary. It can be seen quite clearly that there is a significant amount of duplication of data in these departments, and this is generally true of file-based systems. Before we discuss the limitations of this approach, it may be useful to understand the terminology used in file-based systems. A file is simply a collection of **records**, which contain logically related **data**. For example, the Student file in Computing contains 3 records, one for each student. Each record contains a logically connected set of one or more **fields**, where each field represents some characteristic of the real-world object that is being modelled. The fields of the Student file represent characteristics of properties, such as student number, name and address.

3.1 Limitations of the File-Based Approach

This brief description of traditional file-based systems should be sufficient to allow us to discuss the limitations of this approach.

Separation and isolation of data When data is isolated in separate files, it is more difficult to access data that should be available. For example, if we want to produce a list of students names studying module 'm1' , we need to access the Modules file to create a temporary list of those students id's who have 'm1' as a module, and then search the Student file for the names that match the student id's recorded. With file-based systems, such processing is difficult.

Duplication of data Due to the decentralised approach taken by each department, the file-based approach encourages the uncontrolled duplication of data. For example, in the two department files above we can clearly see that there is duplication of both student and staff details in the Administration and Computing Departments. Uncontrolled duplication of data is undesirable for several reasons:-

- Duplication is wasteful. It costs time and money to enter the data more than once. Furthermore, it takes up additional storage space, again with associated costs.
- Perhaps more important, duplication can lead to loss of **data integrity**; in other words, the data is no longer consistent. For example, consider the duplication of data between the Administration and Computing Departments above. If a student

changes address and the change of address is communicated only to Computing and not to Administration, the student's administration correspondences will be sent to the wrong address. This example illustrates inconsistencies that may result from the duplication of data. As there is no automatic way for Computing to update the data in the Administration files, it is not difficult to foresee such inconsistencies arising. Even if Administration is notified of the changes, it is still possible that the data may be entered incorrectly.

Data dependence As we have already mentioned, the physical structure and storage of the data files and records are defined in the application code. This means that changes to an existing structure are difficult. For example, increasing the size of the Student file address field from say 40 to 41 characters sounds like a simple change, but it requires the creation of a one-off program (that is, a program that is run only once and can then be discarded) that converts the Student file to the new format. This program has to:-

- Open the original Student file for reading.
- Open a temporary file with the new structure.
- Read a record from the original file, convert the data to conform to the new structure and write it to the temporary file. Repeat for all records in the original file.
- Delete the original Student file.
- Rename the temporary file as Student.

In addition, all programs that access the Student file must be modified to conform to the new file structure. There might be many such programs and the programmer needs to identify all the affected programs, modify them and then retest them. This could be very time-consuming and subject to error. This characteristic of file-based systems is known as **program-data dependence**.

Incompatible file formats As the structure of files is embedded in the application programs, the structure is dependent on the application programming language. For example, the structure of a file generated by a COBOL program may be different from the structure of a file generated by a C program. The direct incompatibility of such files makes them difficult to process jointly.

4 Database System Model

All the above limitations of the file-based approach led to an alternative approach for information systems – the database approach. Figure 1 is a model of a database system. The DBMS software plays a central role in the model with user processes, the database and the data dictionary. A DBMS is essential to the database approach. Often the term a database system is used to describe a database managed by a DBMS. Here we will describe each of the components in the database model.

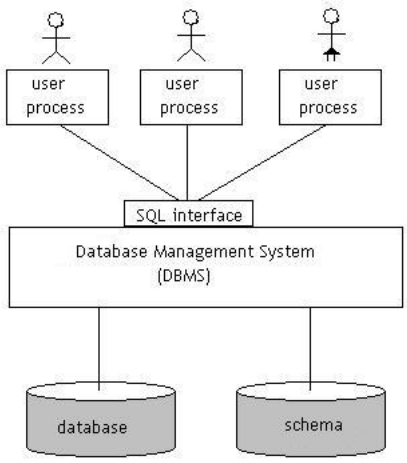

Figure 1: Database System Model

4.1 The Database

A database is a shared collection of logically related data designed to meet the information needs of an organisation. The database is a single, large repository of data, which is defined once and used simultaneously by many users. All data is integrated with a minimum amount of duplication. The database is a shared corporate resource.

Critically, the database consists of tables containing records. For example, a LYIT college database would have a Student table with a record for each student enrolled.

4.2 Schema

Another aspect of the model in Figure 1 is a separate database definition. This special kind of data definition is known as a **schema**, and is an essential part of any database system. A schema contains a specification of the properties of all the data in the associated database tables, and is used by the DBMS to determine how the data is to be processed. The definition of each table gives the data type and other details for each table column, enabling the DBMS to store and retrieve data for that table.

The important point about an explicit schema is not that it is distinct from the database but that it is independent of both a DBMS and user processes and thus data definitions are not embedded in programs, which overcome one of the problems we described for the file-based approach – **program-data dependence**.

You should appreciate a database system consists of a DBMS and a schema, even though they are generally hidden from the normal user. The model in Figure 1 is applicable to any DBMS. This general model is the basis for all database systems like Microsoft Access, Microsoft SQL Server, MySQL, Oracle, IBMs DB2, etc.

4.3 Database Management System (DBMS)

A DBMS is a software system that enables users to define, create and maintain the database and which provides controled access to this database.

In the database model the DBMS is the software that :-

- supports and controls access to a database for every user process;
- manages the storage and retrieval of data in the database;
- processes data according to the schema.

Typically, a DBMS provides the following facilities:

- It allows users to define the database schema, usually through a **Data Definition Language (DDL)** (e.g. SQL). The DDL allows users to specify the data types and structures, and the constraints on the data to be stored in the database.
- It allows users to insert, update, delete and retrieve data from the database, usually through a **Data Manipulation Language (DML)** (e.g. SQL). Having a central repository for all data and data descriptions allows the DML to provide a general enquiry facility to this data, called a query language. The most common query language is the **SQL (Structured Query Language)** pronounced 'S-Q-L' or sometimes 'See-Quel'. It is now both the standard and the de facto language for relational database systems. The spring semester module SQL [SQLACO601] is devoted to the this language. We will look at SQL briefly later in this module.
- It provides controlled access to the database which prevents unauthorised users from accessing the database.

5 DBMS Environment

5.1 Components of the DBMS Environment

We can identify five major components in the DBMS environment – hardware, software, data, procedures and people, as illustrated in Figure 2.

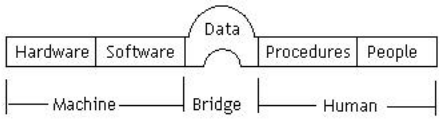

Figure 2: DBMS Environment Components

Hardware The DBMS and the applications require hardware to run. The hardware can range from a single personal computer to a single mainframe to a network of computers. The particular hardware depends on the organisation's requirements and the DBMS used. Some DBMSs run only on particular hardware or operating systems, while others run on a wide variety of hardware and operating systems. A DBMS requires a minimum amount of main memory and disk space to run, but this minimum configuration may not necessarily give acceptable performance.

Software The software component comprises the DBMS software itself together with the operating system, including network software if the DBMS is being used over a network, and the application programs. Typically, application programs are written in programming languages such as C, COBOL, or Java, often using SQL embedded commands.

Data Perhaps the most important component of the DBMS environment, certainly from the end users' point of view, is the data.

Procedures Procedures refer to the instructions and rules that govern the design and use of the database. The users of the system and the staff that manage the database require documented procedures on how to use or run the system. These may consist of instructions on how to:-

- Log on to the DBMS.
- Use a particular DBMS facility or application program.
- Start and stop the DBMS.
- Make backup copies of the database.
- Handle hardware or software failures.

People The final component is the people involved and their role with the system. We discuss this next.

5.2 Roles in the Database Environment

In this section, we examine the fifth component of the DBMS environment – the people. We can identify four distinct types of people – database administrators, database designers, application programmers and end-users.

5.2.1 Data and Database Administrators

The database and the DBMS are corporate resources that must be managed like any other resource. Database administration are the roles generally associated with the management and control of a DBMS and its data.

The **Data Administrator (DA)** is responsible for the management of the data resource including database planning, development and maintenance of standards, policies and procedures and logical database design. The DA consults with and advises senior managers,

ensuring that the direction of database development will ultimately support corporate objectives.

The **Database Administrator (DBA)** is responsible for the physical implementation of the database system, including physical database design and implementation, security control and ensuring satisfactory performance for the applications and users. The role of the DBA is more technically oriented than the role of the DA, requiring detailed knowledge of the target DBMS and the system environment.

In some organisations there is no distinction between these two roles.

5.2.2 Database Designers

In large database design projects, we can distinguish between two types of designers – logical database designers and physical database designers. The **logical database designer** is concerned with identifying the data (that is, the entities and attributes), the relationships between the data and the constraints on the data that is to be stored in the database. The logical database designer must have a thorough and complete understanding of the organisation's data and of the **business rules**. Business rules describe the main characteristics of the data as viewed by the organisation. For example:-

- A student cannot be enrolled on more than one course;
- A student cannot be taught by one of their immediate family members;
- A student must have a CAO number.

The **physical database designer** takes the logical data model and decides how it is to be physically realised. This involves:-

- mapping the logical data model into a set of tables and integrity constraints;
- selecting specific storage structures for the data to achieve good performance for the database activities;
- providing security measures required on the data.

Many parts of physical database design are highly dependent on the actual DBMS used. Consequently, the physical database designer must be fully aware of the functionality of the actual DBMS.

5.2.3 Application Programmers

Once the database has been implemented, the application programs that provide the required functionality for the end users must be implemented. This is the responsibility of the **application programmers**. Typically, the application programmers work from a specification produced by systems analysts. Each program contains statements that request the DBMS to perform some operation on the database. This includes retrieving data, inserting, updating and deleting data. The programs may be written in a programming language such as C, Java, COBOL, etc.

5.2.4 End-Users

The end-users are the *clients* for the database – the database has been designed and implemented, and is being maintained to serve their information needs. End-users can be classified according to the way they use the system:-

Naive users are typically unaware of the DBMS. They access the database through specially written application programs, which attempt to make the operations as simple as possible. They invoke database operations by entering simple commands or choosing options from a menu. This means that they do not need to know anything about the database or the DBMS. For example, the checkout assistant at the local supermarket uses a bar code reader to find out the price of the item.

Sophisticated users are familiar with the structure of the database and the facilities offered by the DBMS. Sophisticated end-users may use SQL to perform the required operations. Some sophisticated end-users may even write application programs for their own use.

6 Advantages & Disadvantages of Databases

In this section, we examine the advantages and disadvantages of the database management system.

6.1 Advantages

6.1.1 Control of data redundancy

Older traditional systems can waste space by storing the same information in more than one file (data redundancy). For example, storing the same contact details for students in both the Computing and Administration Departments. In contrast, the database approach attempts to eliminate the redundancy by integrating the files so that several copies of the same data are not stored. However, the database approach does not eliminate redundancy entirely, but controls the amount of redundancy inherent in the database.

6.1.2 Data consistency

By eliminating or controlling redundancy, we are reducing the risk of inconsistencies occurring. If a data item is stored only once in the database, any update to its value has to be performed only once and the new value is immediately available to all users. If a data item is stored more than once and the system is aware of this, the system can ensure that all copies of the item are kept consistent. Unfortunately, many of today's DBMSs do not automatically ensure this type of consistency.

6.1.3 Sharing of data

Typically, files are owned by the people or departments that use them. On the other hand, the database belongs to the entire organisation and can be shared by all authorised users. In this way, more users share more of the data.

6.1.4 Improved data integrity

Database integrity refers to the validity and consistency of stored data. Integrity is usually expressed in terms of **constraints**, which are consistency rules that the database is not permitted to violate. Constraints may apply to data items within a single record or they may apply to relationships between records. For example, an integrity constraint could state that an student cannot be doing more than six modules in any semester or that the course number contained in the student's record, representing the course the student is enrolled on, corresponds to an existing course offered. Again, integration allows the DBA to define, and the DBMS to enforce, integrity constraints.

6.1.5 Improved security

Database security is the protection of the database from unauthorised users. Without suitable security measures, data may be vulnerable. However, integration allows the DBA to define, and the DBMS to enforce, database security. This may take the form of user names and passwords to identify people authorised to use the database. The access that an authorised user is allowed on the data may be restricted by the operation type (retrieval, insert, update, delete). For example, the DBA has access to all the data in the database; a head of department may have access to all data that relates to his or her department office; and a lecturer may have access to all data relating to academic properties of students but no access to sensitive data, such as contact and grant details.

6.1.6 Economy of scale

Combining all of an organisation's operational data into one database with the applications that are required can result in cost savings. In the case of a college, the budget that would normally be allocated to each department for the development and maintenance of their database systems can be combined, possibly resulting in a lower total cost, leading to an economy of scale. The combined budget can be used to buy a system configuration that is more suited to the organisation's needs. This may consist of one large, powerful computer or a network of smaller computers.

6.1.7 Increased productivity

The DBMS provides many of the standard functions that the application programmer would normally have to write in a file-based applications. At a basic level, the DBMS provides all the low-level file-handling routines that are typical in application programs. The provision

of these functions allows the programmer to concentrate more on the specific functionality required by the users without having to worry about low-level implementation details. This results in increased programmer productivity and reduced development time (with associated cost savings).

6.1.8 Improved backup and recovery services

In a database environment, the database records are normally backed up (copied) on a regular basis. A tape or disk is used to keep the backup secure. As transactions are performed, any updates are recorded in a log of changes. If the system fails, the tape and the log are used to bring the database to the state it was in before the failure. The sysem is therefore self recovering.

6.2 Disadvantages

6.2.1 Complexity

The provision of the functionality we expect of a good DBMS makes the DBMS an extremely complex piece of software. Database designers and developers, the data and database administrators and end-users must understand this functionality to take full advantage of it.

6.2.2 Size

The complexity and breadth of functionality makes the DBMS an extremely large piece of software, occupying many megabytes of disk space and requiring substantial amounts of memory to run efficiently.

6.2.3 Cost of DBMS

The cost of DBMSs varies significantly, depending on the environment and functionality provided. For example, a single-user DBMS (e.g. Microsoft Access) for a personal computer may only cost £500. However, a large mainframe multi-user DBMS servicing hundreds of users can be extremely expensive, perhaps £100,000 to £500,000. Plus, there is also the recurrent annual maintenance cost.

6.2.4 Additional hardware costs

The disk storage requirements for the DBMS and the database may necessitate the purchase of additional storage space. Furthermore, to achieve the required performance, it may be necessary to purchase a larger machine with a fast processor and more memory, perhaps even a machine dedicated to running the DBMS. The procurement of additional hardware results in further expenditure.

6.2.5 Cost of conversion

In some situations, the cost of the DBMS and extra hardware may be insignificant compared with the cost of converting existing applications to run on/with the new DBMS and hardware. Coupled with this is the cost of having to transfer (or convert) data from the old system to the new one. Another cost includes the cost of training staff to use these new systems, and possibly the employment of specialist staff to help with the conversion and running of the system. These costs are one of the main reasons why some organisations feel tied to their current legacy systems and cannot switch to newer database technology.

6.2.6 Performance

Typically, a file-based system is written for a specific application, such as invoicing. As a result, performance is generally very good. However, the DBMS is written to be more general, to cater for many applications rather than just one. The effect is that some applications may not run as fast any more.

6.2.7 Higher impact of a failure

The centralisation of resources increases the vulnerability of the system. Since all users and applications rely on the availability of the DBMS, the failure of any component can bring operations to a halt.

References

[1] Catherine M. Ricardo. *Databases Illuminated*. Jones and Bartlett, 2004.

[2] Carolyn Begg Thomas Connolly and Anne Strachan. *Database Systems*. Addison-Wesley, 1997.