Chapter 1. Introduction to the Module

Table of contents

The purpose of this chapter is to introduce the fundamental concepts of database systems. Like most areas of computing, database systems have a significant number of terms and concepts that are likely to be new to you. We encourage you to discuss these terms in tutorials and online with one another, and to share any previous experience of database systems that you may have. The module covers a wide range of issues associated with database systems, from the stages and techniques used in the development of database applications, through to the administration of complex database environments. The overall aim of the module is to equip you with the knowledge required to be a valuable

member of a team, or to work individually, in the areas of database application development or administration. In addition, some coverage of current research areas is provided, partly as a stimulus for possible future dissertation topics, and also to provide an awareness of possible future developments within the database arena.

Module objectives

At the end of this module you will have acquired practical and theoretical knowledge and skills relating to modern database systems. The module is designed so that this knowledge will be applicable across a wide variety of database environments. At the end of the module you will be able to:

- Understand and explain the key ideas underlying database systems and the database approach to information storage and manipulation.
- Design and implement database applications.
- Carry out actions to improve the performance of existing database applications.
- Understand the issues involved in providing multiple users concurrent access to database systems.
- Be able to design adequate backup, recovery and security measures for a database installation, and understand the facilities provided by typical database systems to support these tasks.
- Understand the types of tasks involved in database administration and the facilities provided in a typical database system to support these tasks.
- Be able to describe the issues and objectives in a range of areas of contemporary database research.

Chapter objectives

At the end of this chapter you should be able to:

- Explain the advantages of a database approach for information storage and retrieval.
- Explain the concepts of physical and logical data independence, and describe both technically and in business terms the advantages that these concepts provide in Information Systems development.
- Understand the basic terminology and constructs of the Relational approach to database systems.

Introduction

In parallel with this chapter, you should read Chapter 1 and Chapter 2 of Thomas Connolly and Carolyn Begg, "Database Systems A Practical Approach to Design, Implementation, and Management", (5th edn.).

This chapter sets the scene for all of the forthcoming chapters of the module. We begin by examining the approach to storing and processing data that was used before the arrival of database systems, and that is still appropriate today in certain situations (which will be explained). We then go on to examine the difference between this traditional, file-based approach to data storage, and that of the database approach. We do this first by examining inherent limitations of the file-based approach, and then discuss ways in which the database approach can be used to overcome these limitations.

A particular model of database systems, known as the Relational model, has been the dominant approach in the database industry since the early '80s. There are now important rivals and extensions to the Relational model, which will be examined in later chapters, but the Relational model remains the core technology on which the database industry worldwide is based, and for this reason this model will be central to the entire module.

Motivation for data storage

Day-to-day business processes executed by individuals and organisations require both present and historical data. Therefore, data storage is essential for organisations and individuals. Data supports business functions and aids in business decision-making. Below are some of the examples where data storage supports business functions.

Social media

Social media has become very popular in the 21st century. We access social media using our computers and mobile phones. Every time we access social media, we interact, collaborate and share content with other people. The owners of social media platforms store the data we produce.

Supermarket

A supermarket stores different types of information about its products, such as quantity, prices and type of product. Every time we buy anything from the supermarket, quantities must be reduced and the sales information must be stored.

Company

A company will need to hold details of its staff, customers, products, suppliers and financial transactions.

If there are a small number of records to be kept, and these do not need to be changed very often, a card index might be all that is required. However, where there is a high volume of data, and a need to manipulate this data on a regular basis, a computer-based solution will often be chosen. This might sound like a simple solution, but there are a number of different approaches that could be taken.

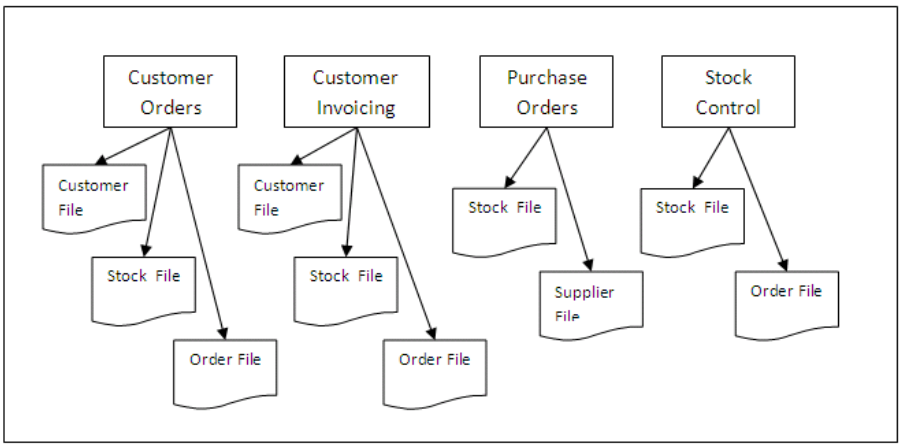
Traditional file-based approach

The term 'file-based approach' refers to the situation where data is stored in one or more separate computer files defined and managed by different application programs. Typically, for example, the details of customers may be stored in one file, orders in another, etc. Computer programs access the stored files to perform the various tasks required by the business. Each program, or sometimes a related set of programs, is called a computer application. For example, all of the programs associated with processing customers' orders are referred to as the order processing application. The file-based approach might have application programs that deal with purchase orders, invoices, sales and marketing, suppliers, customers, employees, and so on.

Limitations

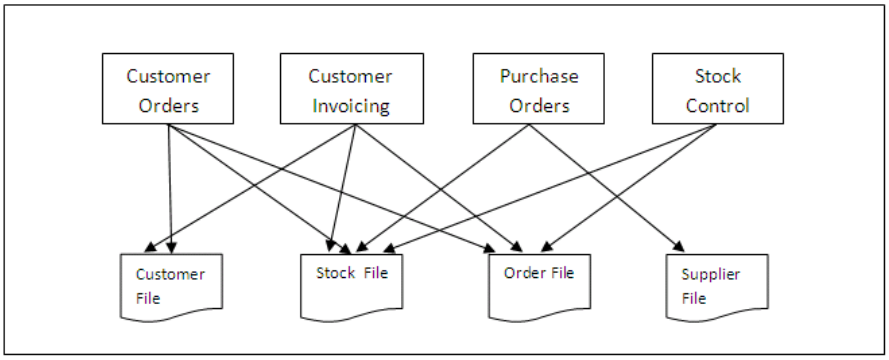
- Data duplication: Each program stores its own separate files. If the same data is to be accessed by different programs, then each program must store its own copy of the same data.
- Data inconsistency: If the data is kept in different files, there could be problems when an item of data needs updating, as it will need to be updated in all the relevant files; if this is not done, the data will be inconsistent, and this could lead to errors.
- Difficult to implement data security: Data is stored in different files by different application programs. This makes it difficult and expensive to implement organisation-wide security procedures on the data.

The following diagram shows how different applications will each have their own copy of the files they need in order to carry out the activities for which they are responsible:



The shared file approach

One approach to solving the problem of each application having its own set of files is to share files between different applications. This will alleviate the problem of duplication and inconsistent data between different applications, and is illustrated in the diagram below:



The introduction of shared files solves the problem of duplication and inconsistent data across different versions of the same file held by different departments, but other problems may emerge, including:

- File incompatibility: When each department had its own version of a file for processing, each department could ensure that the structure of the file suited their specific application. If departments have to share files, the file structure that suits one department might not suit another. For

example, data might need to be sorted in a different sequence for different applications (for instance, customer details could be stored in alphabetical order, or numerical order, or ascending or descending order of customer number).

- Difficult to control access: Some applications may require access to more data than others; for instance, a credit control application will need access to customer credit limit information, whereas a delivery note printing application will only need access to customer name and address details. The file will still need to contain the additional information to support the application that requires it.
- Physical data dependence: If the structure of the data file needs to be changed in some way (for example, to reflect a change in currency), this alteration will need to be reflected in all application programs that use that data file. This problem is known as physical data dependence, and will be examined in more detail later in the chapter.
- Difficult to implement concurrency: While a data file is being processed by one application, the file will not be available for other applications or for ad hoc queries. This is because, if more than one application is allowed to alter data in a file at one time, serious problems can arise in ensuring that the updates made by each application do not clash with one another. This issue of ensuring consistent, concurrent updating of information is an extremely important one, and is dealt with in detail for database systems in the chapter on concurrency control. File-based systems avoid these problems by not allowing more than one application to access a file at one time.

Review question 1

What is meant by the file-based approach to storing data? Describe some of the disadvantages of this approach.

Review question 2

How can some of the problems of the file-based approach to data storage be avoided?

Review question 3

What are the problems that remain with the shared file approach?

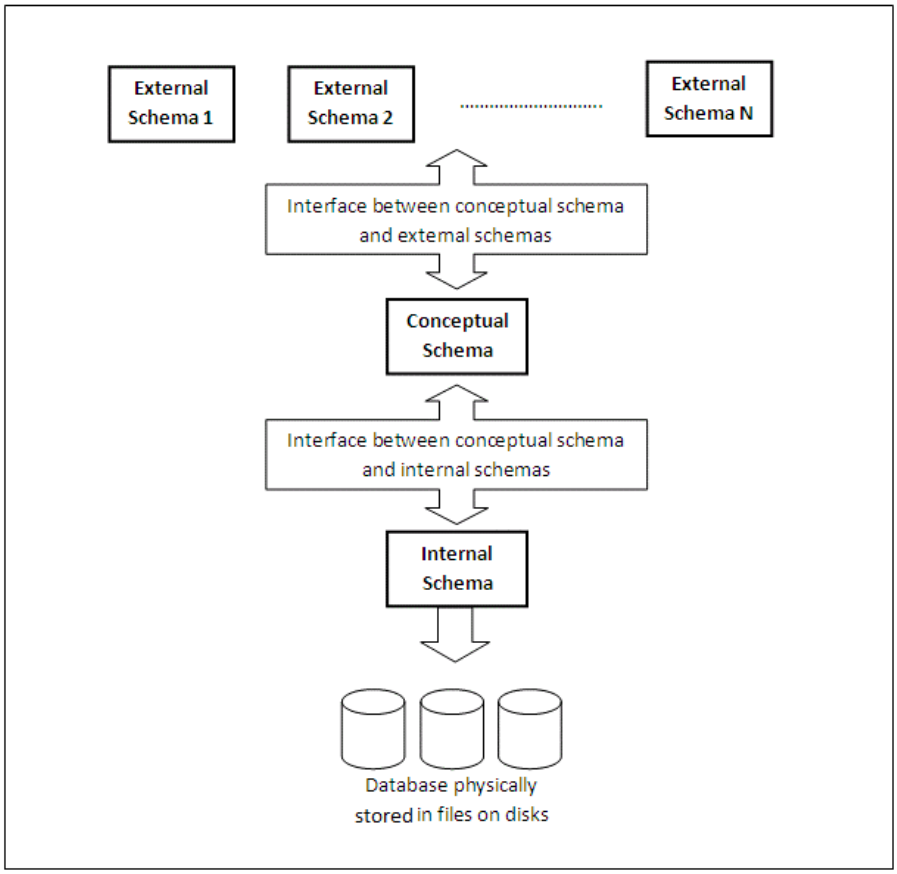
The database approach

The database approach is an improvement on the shared file solution as the use of a database management system (DBMS) provides facilities for querying, data security and integrity, and allows simultaneous access to data by a number of different users. At this point we should explain some important terminology:

- **Database:** A database is a collection of related data.
- **Database management system:** The term 'database management system', often abbreviated to DBMS, refers to a software system used to create and manage databases. The software of such systems is complex, consisting of a number of different components, which are described later in this chapter. The term database system is usually an alternative term for database management system.
- **System catalogue/Data dictionary:** The description of the data in the database management system.
- **Database application:** Database application refers to a program, or related set of programs, which use the database management system to perform the computer-related tasks of a particular business function, such as order processing.

One of the benefits of the database approach is that the problem of physical data dependence is resolved; this means that the underlying structure of a data file can be changed without the application programs needing amendment. This is achieved by a hierarchy of levels of data specification. Each such specification of data in a database system is called a schema. The different levels of schema provided in database systems are described below. Further details of what is included within each specific schema are discussed later in the chapter.

The Systems Planning and Requirements Committee of the American National Standards Institute encapsulated the concept of schema in its three-level database architecture model, known as the ANSI/SPARC architecture, which is shown in the diagram below:



ANSI/SPARC three-level architecture

ANSI = American National Standards Institute

ANSI/X3 = Committee on Computers and Information Processing

SPARC = Standards Planning and Requirements Committee

The ANSI/SPARC model is a three-level database architecture with a hierarchy of levels, from the users and their applications at the top, down to the physical storage of data at the bottom. The characteristics of each level, represented by a schema, are now described.

The external schema

The external schemas describe the database as it is seen by the user, and the

user applications. The external schema maps onto the conceptual schema, which is described below.

There may be many external schemas, each reflecting a simplified model of the world, as seen by particular applications. External schemas may be modified, or new ones created, without the need to make alterations to the physical storage of data. The interface between the external schema and the conceptual schema can be amended to accommodate any such changes.

The external schema allows the application programs to see as much of the data as they require, while excluding other items that are not relevant to that application. In this way, the external schema provides a view of the data that corresponds to the nature of each task.

The external schema is more than a subset of the conceptual schema. While items in the external schema must be derivable from the conceptual schema, this could be a complicated process, involving computation and other activities.

The conceptual schema

The conceptual schema describes the universe of interest to the users of the database system. For a company, for example, it would provide a description of all of the data required to be stored in a database system. From this organisation-wide description of the data, external schemas can be derived to provide the data for specific users or to support particular tasks.

At the level of the conceptual schema we are concerned with the data itself, rather than storage or the way data is physically accessed on disk. The definition of storage and access details is the preserve of the internal schema.

The internal schema

A database will have only one internal schema, which contains definitions of the way in which data is physically stored. The interface between the internal schema and the conceptual schema identifies how an element in the conceptual schema is stored, and how it may be accessed.

If the internal schema is changed, this will need to be addressed in the interface between the internal and the conceptual schemas, but the conceptual and external schemas will not need to change. This means that changes in physical storage devices such as disks, and changes in the way files are organised on storage devices, are transparent to users and application programs.

In distinguishing between 'logical' and 'physical' views of a system, it should be noted that the difference could depend on the nature of the user. While 'logical' describes the user angle, and 'physical' relates to the computer view, database designers may regard relations (for staff records) as logical and the database

itself as physical. This may contrast with the perspective of a systems programmer, who may consider data files as logical in concept, but their implementation on magnetic disks in cylinders, tracks and sectors as physical.

Physical data independence

In a database environment, if there is a requirement to change the structure of a particular file of data held on disk, this will be recorded in the internal schema. The interface between the internal schema and the conceptual schema will be amended to reflect this, but there will be no need to change the external schema. This means that any such change of physical data storage is not transparent to users and application programs. This approach removes the problem of physical data dependence.

Logical data independence

Any changes to the conceptual schema can be isolated from the external schema and the internal schema; such changes will be reflected in the interface between the conceptual schema and the other levels. This achieves logical data independence. What this means, effectively, is that changes can be made at the conceptual level, where the overall model of an organisation's data is specified, and these changes can be made independently of both the physical storage level, and the external level seen by individual users. The changes are handled by the interfaces between the conceptual, middle layer, and the physical and external layers.

Review question 4

What are some of the advantages of the database approach compared to the shared file approach of storing data?

Review question 5

Distinguish between the terms 'external schema', 'conceptual schema' and 'internal schema'.

Components of a DBMS

The major components of a DBMS are as follows:

DBMS engine

The engine is the central component of a DBMS. This component provides access to the database and coordinates all of the functional elements of the DBMS. An important source of data for the DBMS engine, and the database system as a whole, is known as metadata. Metadata means data about data. Metadata is

contained in a part of the DBMS called the data dictionary (described below), and is a key source of information to guide the processes of the DBMS engine. The DBMS engine receives logical requests for data (and metadata) from human users and from applications, determines the secondary storage location (i.e. the disk address of the requested data), and issues physical input/output requests to the computer operating system. The data requested is fetched from physical storage into computer main memory; it is contained in special data structures provided by the DBMS. While the data remains in memory, it is managed by the DBMS engine. Additional data structures are created by the database system itself, or by users of the system, in order to provide rapid access to data being processed by the system. These data structures include indexes to speed up access to the data, buffer areas into which particular types of data are retrieved, lists of free space, etc. The management of these additional data structures is also carried out by the DBMS engine.

User interface subsystem

The interface subsystem provides facilities for users and applications to access the various components of the DBMS. Most DBMS products provide a range of languages and other interfaces, since the system will be used both by programmers (or other technical persons) and by users with little or no programming experience. Some of the typical interfaces to a DBMS are the following:

- A data definition language (or data sublanguage), which is used to define, modify or remove database structures such as records, tables, files and views.
- A data manipulation language, which is used to display data extracted from the database and to perform simple updates and deletions.
- A data control language, which allows a database administrator to have overall control of the system, often including the administration of security, so that access to both the data and processes of the database system can be controlled.
- A graphical user interface, which may provide a visual means of browsing or querying the data, including a range of different display options such as bar charts, pie charts, etc. One particular example of such a system is Query-by-Example, in which the system displays a skeleton table (or tables), and users pose requests by suitable entry in the table.
- A forms-based user interface in which a screen-oriented form is presented to the user, who responds by filling in blanks on the form. Such forms-based systems are a popular means of providing a visual front-end to both developers and users of a database system. Typically, developers use the forms-based system in 'developer mode', where they design the forms or screens that will make up an application, and attach fragments of code

which will be triggered by the actions of users as they use the forms-based user interface.

- A DBMS procedural programming language, often based on standard third-generation programming languages such as C and COBOL, which allows programmers to develop sophisticated applications.
- Fourth-generation languages, such as Smalltalk, JavaScript, etc. These permit applications to be developed relatively quickly compared to the procedural languages mentioned above.
- A natural language user interface that allows users to present requests in free-form English statements.

Data dictionary subsystem

The data dictionary subsystem is used to store data about many aspects of how the DBMS works. The data contained in the dictionary subsystem varies from DBMS to DBMS, but in all systems it is a key component of the database. Typical data to be contained in the dictionary includes: definitions of the users of the system and the access rights they have, details of the data structures used to contain data in the DBMS, descriptions of business rules that are stored and enforced within the DBMS, and definitions of the additional data structures used to improve systems performance. It is important to understand that because of the important and sensitive nature of the data contained in the dictionary subsystem, most users will have no or little direct access to this information. However, the database administrator will need to have regular access to much of the dictionary system, and should have a detailed knowledge of the way in which the dictionary is organised.

Performance management subsystem

The performance management subsystem provides facilities to optimise (or at least improve) DBMS performance. This is necessary because the large and complex software in a DBMS requires attention to ensure it performs efficiently, i.e. it needs to allow retrieval and changes to data to be made without requiring users to wait for significant periods of time for the DBMS to carry out the requested action.

Two important functions of the performance management subsystem are:

- Query optimisation: Structuring SQL queries (or other forms of user queries) to minimise response times.
- DBMS reorganisation: Maintaining statistics on database usage, and taking (or recommending) actions such as database reorganisation, creating indexes and so on, to improve DBMS performance.

Data integrity management subsystem

The data integrity management subsystem provides facilities for managing the integrity of data in the database and the integrity of metadata in the dictionary. This subsystem is concerned with ensuring that data is, as far as software can ensure, correct and consistent. There are three important functions:

- Intra-record integrity: Enforcing constraints on data item values and types within each record in the database.
- Referential integrity: Enforcing the validity of references between records in the database.
- Concurrency control: Ensuring the validity of database updates when multiple users access the database (discussed in a later chapter).

Backup and recovery subsystem

The backup and recovery subsystem provides facilities for logging transactions and database changes, periodically making backup copies of the database, and recovering the database in the event of some type of failure. (We discuss backup and recovery in greater detail in a later chapter.) A good DBMS will provide comprehensive and flexible mechanisms for backing up and restoring copies of data, and it will be up to the database administrator, in consultation with users of the system, to decide precisely how these features should be used.

Application development subsystem

The application development subsystem is for programmers to develop complete database applications. It includes CASE tools (software to enable the modelling of applications), as well as facilities such as screen generators (for automatically creating the screens of an application when given details about the data to be input and/or output) and report generators.

In most commercial situations, there will in fact be a number of different database systems, operating within a number of different computer environments. By computer environment we mean a set of programs and data made available usually on a particular computer. One such set of database systems, used in a number of medium to large companies, involves the establishment of three different computer environments. The first of these is the development environment, where new applications are developed and new applications, whether written within the company or bought in from outside, are tested. The development environment usually contains relatively little data, just enough in fact to adequately test the logic of the applications being developed and tested. Security within the development environment is usually not an important issue, unless the actual logic of the applications being developed is, in its own right, of a sensitive nature.

The second of the three environments is often called pre-production. Applications that have been tested in the development environment will be moved into pre-production for volume testing; that is, testing with quantities of data that are typical of the application when it is in live operation.

The final environment is known as the production or live environment. Applications should only be moved into this environment when they have been fully tested in pre-production. Security is nearly always a very important issue in the production environment, as the data being used reflects important information in current use by the organisation.

Each of these separate environments will have at least one database system, and because of the widely varying activities and security measures required in each environment, the volume of data and degree of administration required will itself vary considerably between environments, with the production database(s) requiring by far the most support.

Given the need for the database administrator to migrate both programs and data between these environments, an important tool in performing this process will be a set of utilities or programs for migrating applications and their associated data both forwards and backwards between the environments in use.

Security management subsystem

The security management subsystem provides facilities to protect and control access to the database and data dictionary.

Benefits of the database approach

The benefits of the database approach are as follows:

- Ease of application development: The programmer is no longer burdened with designing, building and maintaining master files.
- Minimal data redundancy: All data files are integrated into a composite data structure. In practice, not all redundancy is eliminated, but at least the redundancy is controlled. Thus inconsistency is reduced.
- Enforcement of standards: The database administrator can define standards for names, etc.
- Data can be shared. New applications can use existing data definitions.
- Physical data independence: Data descriptions are independent of the application programs. This makes program development and maintenance an easier task. Data is stored independently of the program that uses it.
- Logical data independence: Data can be viewed in different ways by different users.

- Better modelling of real-world data: Databases are based on semantically rich data models that allow the accurate representation of real-world information.
- Uniform security and integrity controls: Security control ensures that applications can only access the data they are required to access. Integrity control ensures that the database represents what it purports to represent.
- Economy of scale: Concentration of processing, control personal and technical expertise.

Risks of the database approach

- New specialised personnel: Need to hire or train new personnel e.g. database administrators and application programmers.
- Need for explicit backup.
- Organisational conflict: Different departments have different information needs and data representation.
- Large size: Often needs alarmingly large amounts of processing power.
- Expensive: Software and hardware expenses.
- High impact of failure: Concentration of processing and resources makes an organisation vulnerable if the system fails for any length of time.

Review question 6

Distinguish between the terms 'database security' and 'data integrity'.

Data and database administration

Organisations need data to provide details of the current state of affairs; for example, the amount of product items in stock, customer orders, staff details, office and warehouse space, etc. Raw data can then be processed to enable decisions to be taken and actions to be made. Data is therefore an important resource that needs to be safeguarded. Organisations will therefore have rules, standards, policies and procedures for data handling to ensure that accuracy is maintained and that proper and appropriate use is made of the data. It is for this reason that organisations may employ data administrators and database administrators.

The role of the data administrator

It is important that the data administrator is aware of any issues that may affect the handling and use of data within the organisation. Data administration

includes the responsibility for determining and publicising policy and standards for data naming and data definition conventions, access permissions and restrictions for data and processing of data, and security issues.

The data administrator needs to be a skilled manager, able to implement policy and make strategic decisions concerning the organisation's data resource. It is not sufficient for the data administrator to propose a set of rules and regulations for the use of data within an organisation; the role also requires the investigation of ways in which the organisation can extract the maximum benefit from the available data.

One of the problems facing the data administrator is that data may exist in a range of different formats, such as plain text, formatted documents, tables, charts, photographs, spreadsheets, graphics, diagrams, multimedia (including video, animated graphics and audio), plans, etc. In cases where the data is available on computer-readable media, consideration needs to be given to whether the data is in the correct format.

The different formats in which data may appear is further complicated by the range of terms used to describe it within the organisation. One problem is the use of synonyms, where a single item of data may be known by a number of different names. An example of the use of synonyms would be the terms 'telephone number', 'telephone extension', 'direct line', 'contact number' or just 'number' to mean the organisation's internal telephone number for a particular member of staff. In an example such as this, it is easy to see that the terms refer to the same item of data, but it might not be so clear in other contexts.

A further complication is the existence of homonyms. A homonym is a term which may be used for several different items in different contexts; this can often happen when acronyms are used. One example is the use of the terms 'communication' and 'networking'; these terms are sometimes used to refer to interpersonal skills, but may also be employed in the context of data communication and computer networks.

When the items of data that are important to an organisation have been identified, it is important to ensure that there is a standard representation format. It might be acceptable to tell a colleague within the organisation that your telephone extension is 5264, but this would be insufficient information for someone outside the organisation. It may be necessary to include full details, such as international access code, national code, area code and local code as well as the telephone extension to ensure that the telephone contact details are usable worldwide.

Dates are a typical example of an item of data with a wide variety of formats. The ranges of date formats include: day-month-year, month-day-year, year-month-day, etc. The month may appear as a value in the range 1 to 12, as the name of the month in full, or a three-letter abbreviation. These formats can be varied by changing the separating character between fields from a hyphen (-) to a slash (/), full stop (.) or space ( ).

The use of standardised names and formats will assist an organisation in making good use of its data. The role of the data administrator involves the creation of these standards and their publication (including the reasons for them and guidelines for their use) across the organisation. Data administration provides a service to the organisation, and it is important that it is perceived as such, rather than the introduction of unnecessary rules and regulations.

The role of the database administrator

The role of the database administrator within an organisation focuses on a particular database or set of databases, and the associated computer applications, rather than the use of data throughout the organisation. A database administrator requires a blend of management skills together with technical expertise. In smaller organisations, the data administrator and database administrator roles may be merged into a single post, whereas larger companies may have groups of staff involved with each activity.

The activities of the database administrator take place in the context of the guidelines set out by the data administrator. This requires striking a balance between the security and protection of the database, which may be in conflict with the requirements of users to have access to the data. The database administrator has responsibility for the development, implementation, operation, maintenance and security of the database and the applications that use it. Another important function is the introduction of controls to ensure the quality and integrity of the data that is entered into the database. The database administrator is a manager of the data in the database, rather than a user. This role requires the development of the database structure and data dictionary (a catalogue of the data in the database), the provision of security measures to permit authorised access and prevent unauthorised access to data, and to guard against failures in hardware or software in order to offer reliability.

Exercise 1

Find out who is responsible for the tasks of data administration and database administration in the organisation where you are currently working or studying. Find out whether the two roles are combined into one in your organisation, or if not, how many people are allocated to each function, and what are their specific roles?

Introduction to the Relational model

A number of different approaches or models have been developed for the logical organisation of data within a database system. This 'logical' organisation must be distinguished from the 'physical' organisation of data, which describes how the data is stored on some suitable storage medium such as a disk. The physical

organisation of data will be dealt with in the chapter on physical storage. By far the most commonly used approach to the logical organisation of data is the Relational model. In this section we shall introduce the basic concepts of the Relational model, and give examples of its use. Later in the module, we shall make practical use of this knowledge in both using and developing examples of Relational database applications.

Entities, attributes and relationships

The first step in the development of a database application usually involves determining what the major elements of data to be stored are. These are referred to as entities. For example, a library database will typically contain entities such as Books, Borrowers, Librarians, Loans, Book Purchases, etc. Each of the entities identified will contain a number of properties, or attributes. For example, the entity Book will contain attributes such as Title, Author and ISBN; the entity Borrower will possess attributes such as Name, Address and Membership Number. When we have decided which entities are to be stored in a database, we also need to consider the way in which those entities are related to one another. Examples of such relationships might be, for the library system, that a Borrower can borrow a number of Books, and that a Librarian can make a number of Book Purchases. The correct identification of the entities and attributes to be stored, and the relationships between them, is an extremely important topic in database design, and will be covered in detail in the chapter on entity-relationship modelling. In introducing the Relational approach to database systems, we must consider how entities and their attributes, and the relationships between them, will be represented within a database system.

A relation is structured like a table. The rows of the structure (which are also sometimes referred to as tuples) correspond to individual instances of records stored in the relation. Each column of the relation corresponds to a particular attribute of those record instances. For example, in the relation containing details of stationery below, each row of the relation corresponds to a different item of stationery, and each column or attribute corresponds to a particular aspect of stationery, such as the colour or price.

Each tuple contains values for a fixed number of attributes. There is only one tuple for each different item represented in the database.

The set of permissible values for each attribute is called the domain for that attribute. It can be seen that the domain for the attribute Colour in the stationery relation below includes the values Red, Blue, Green, White, Yellow, and Black (other colours may be permitted but are not shown in the relation).

The sequence in which tuples appear within a relation is not important, and the order of attributes within a relation is of no significance. However, once the attributes of a particular relation have been identified, it is convenient to refer to them in the same order.

Very often it is required to be able to identify uniquely each of the different instances of entities in a database. In order to do this we use something called a primary key. We will discuss the nature of primary keys in detail in the next learning chapter, but for now we shall use examples where the primary key is the first of the attributes in each tuple of a relation.

Relation: Stationery

Item-code	Item-name	Colour	Price
19876	A4 folder	Red	0.25
19877	A4 folder	Blue	0.25
19878	A4 folder	Green	0.25
20216	A4 paper 250 sheets	Red	2.75
20217	A4 paper 250 sheets	Blue	2.75
20218	A4 paper 250 sheets	Green	2.75
20219	A4 paper 250 sheets	White	2.50
33006	A4 ring binder	Red	1.95
33007	A4 ring binder	Green	1.95
33008	A4 ring binder	Blue	1.95
33010	A4 ring binder	Yellow	1.95
33015	A4 ring binder	Black	1.50

Here, the attributes are item-code, item-name, colour and price. The values for each attribute for each item are shown as a single value in each column for a particular row. Thus for item-code 20217, the values are A4 paper 250 sheets for the item-name, Blue for the attribute colour, and <=2.75 is stored as the price.

Question: Which of the attributes in the stationery relation do you think would make a suitable key, and why?

The schema defines the 'shape' or structure of a relation. It defines the number of attributes, their names and domains. Column headings in a table represent the schema. The extension is the set of tuples that comprise the relation at any time. The extension (contents) of a relation may vary, but the schema (structure) generally does not.

From the example above, the schema is represented as:

Item-code	Item-name	Colour	Price

The extension from the above example is given as:

19876	A4 folder	Red	0.25
19877	A4 folder	Blue	0.25
19878	A4 folder	Green	0.25
20216	A4 paper 250 sheets	Red	2.75
20217	A4 paper 250 sheets	Blue	2.75
20218	A4 paper 250 sheets	Green	2.75
20219	A4 paper 250 sheets	White	2.50
33006	A4 ring binder	Red	1.95
33007	A4 ring binder	Green	1.95
33008	A4 ring binder	Blue	1.95
33010	A4 ring binder	Yellow	1.95
33015	A4 ring binder	Black	1.50

The extension will vary as rows are inserted or deleted from the table, or values of attributes (e.g. price) change. The number of attributes will not change, as this is determined by the schema. The number of rows in a relation is sometimes referred to as its cardinality. The number of attributes is sometimes referred to as the degree or grade of a relation.

Each relation needs to be declared, its attributes defined, a domain specified for each attribute, and a primary key identified.

Review question 7

Distinguish between the terms 'entity' and 'attribute'. Give some examples of entities and attributes that might be stored in a hospital database.

Review question 8

The range of values that a column in a relational table may be assigned is called the domain of that column. Many database systems provide the possibility of specifying limits or constraints upon these values, and this is a very effective way of screening out incorrect values from being stored in the system. It is useful, therefore, when identifying which attributes or columns we wish to store for an entity, to consider carefully what is the domain for each column, and which values are permissible for that domain.

Consider then for the following attributes, what the corresponding domains are,

and whether there are any restrictions we can identify which we might use to validate the correctness of data values entered into attributes with each domain:

- Attribute: EMPLOYEE_NAME
- Attribute: JOB (i.e. the job held by an individual in an organisation)
- Attribute: DATE_OF_BIRTH

Discussion topic

External schemas can be used to give individual users, or groups of users, access to a part of the data in a database. Many systems also allow the format of the data to be changed for presentation in the external schema, or for calculations to be carried out on it to make it more usable to the users of the external schema. Discuss the possible uses of external schemas, and the sorts of calculations and/or reformatting that might be used to make the data more usable to specific users or user groups.

External schemas might be used to provide a degree of security in the database, by making available to users only that part of the database that they require in order to perform their jobs. So for example, an Order Clerk may be given access to order information, while employees working in Human Resources may be given access to the details of employees.

In order to improve the usability of an external schema, the data in it may be summarised or organised into categories. For example, an external schema for a Sales Manager, rather than containing details of individual sales, might contain summarised details of sales over the last six months, perhaps organised into categories such as geographical region. Furthermore, some systems provide the ability to display data graphically, in which case it might be formatted as a bar, line or pie chart for easier viewing.

Additional content and activities

Database systems have become ubiquitous throughout computing. A great deal of information is written and published describing advances in database technology, from research papers through to tutorial information and evaluations of commercial products. Conduct a brief search on the Internet and related textbooks. You will likely find that there are many alternative definitions and explanations to the basic concepts introduced in this chapter, and these will be helpful in consolidating the material covered here.