9

Database

RORY R. MCNEILL

Basic database concepts

Database is a term that has entered common language as a result of the information revolution and the age of technology. As such, the term is often used incorrectly, even abused, and has taken on many connotations quite different from its original meaning in the information systems domain.

Simple facts by themselves are of little value. When these facts are organised into a consistent, flexible framework they can be used to prove or disprove an hypothesis, substantiate or refute a claim, and indicate preferable alternatives for decision makers.

In a broad sense, a database is such a collection of simple facts, or data. The value in a database comes from the organisation of the data as much as from the data itself. This view of a database can include both manual data storage (such as an organised filing cabinet), and data storage on a computer.

For our purposes we will take the narrower view of databases and require that they be based on computer-accessible media. Furthermore, given the emphasis on organisation, this discussion is restricted to databases that are part of a database management system: a set of computer programs and files designed for the management (acquisition, organisation, storage, distribution, use, protection, archiving, and removal) of a data resource. Such a system is customarily purchased as a package with a number of options from one major software vendor. The features of these systems are discussed in the next section. The common acronym for database management system is DBMS.

Typical features of database management systems

Most, but not all, database management systems will exhibit all of the following features. A database management system consists of a set of data (database) stored on files, a data dictionary describing the format, structure, and location of the data stored, a user interface allowing interactive query and update to the database, a set of utility programs for monitoring what is happening on the database, re-organising the database, backing up and recovering the database, and an application program interface allowing programs written in one of many computer languages to access the database. Slightly less common features include a system for managing concurrent use of the system by many people, security and access control

features, and higher level languages suitable for developing entire applications. A system is not complete without the people involved with it, in the case of a database management system we usually see a database administrator, software developers, and end-users. A database management system rarely allows access to the data by any means other than its own software. In many cases this software implements a security system restricting access to authorised use. By forcing programs to use the database management system software, data independence is achieved. Changes to the database may be made with only minimal changes required to application programs. Database management systems are often concerned with the sharing of data, this requires the ability to manage many users accessing the database at the same time. Database management systems often provide protection against the failure of the system, with periodic backups made automatically, and with recording of all activity that changes values in the database.

What is it about database management systems that make them important?

Database management systems are perhaps the primary mechanism for getting control of the organisation's information resource. By design, the data within a database can be shared, it has a logical, meaningful structure, and it can support the strategic and tactical goals of the organisation.

Data analysis and modelling

Even if the data needed for a new strategic thrust of the organisation is already captured by some part of the organisation, it is not likely to be of much use unless it has a form and structure that can be used by other parts. To ensure that this form and structure is correct, new and re-written applications must take a close look at the underlying reality. If the data model used is an accurate representation of this reality, the chances are good that the data will be able to be used in the future in ways that cannot be foreseen. Data analysis is the practice of examining the data and information requirements of an information system, and using one or models to represent these requirements in a systematic way. The most common approaches for modelling database requirements are the Entity Relationship data model and the Object-Oriented data model.

Why model? - The modelling process

A model is an abstraction of reality, representing those features that we consider important for the purposes of the model, and ignoring those that we consider unimportant. But what is the purpose of a model? By focusing on the important features, the model helps us to a better understanding of the underlying reality. We can use the model to make generalisations, to categorise and classify, to simplify, to organise, and to predict. Models help us to clarify our grasp of the subject modelled, and help us communicate our ideas to other people.

Models can take many forms. They could be clay models intended to show the shape of a novel aircraft, and used in wind-tunnels to test the aerodynamic characteristics of the proposed shape. They could be dummies with realistic facial features, mouth construction and body size, used to teach CPR techniques when practicing on live people would be dangerous and unsanitary. They could be financial models in the form of spreadsheets, where the interaction of many financial variables can be estimated and the results used to help decide for or against investing in a project. They could be a large set of mathematical

equations, used to understand and predict the behaviour of atoms, molecules, or galaxies under certain circumstances. They could be anatomical models in the form of diagrams and charts, used to teach the identification of body parts and functions.

Models can be static, used to represent existing form, structure, and relationships, or they can be dynamic, used to represent effects and interactions over a period of time.

Models may be created and used once for their purpose, or they may be reused time and time again. Models may be permanent, reflecting the area of reality well enough for continued use, or they may change and evolve as the understanding of the underlying reality changes and evolves.

Models are often composite or layered. A composite model may consist of a number of interrelated sub-models, each of which highlights a different aspect of the reality being modelled. A layered model has a sequence of similar models, with the top level showing only the most significant features and generalities, and the subsequent levels, each introducing features and details less important than shown on higher levels.

The modelling process begins when somebody realises there is some facet of reality that they need to understand better or to communicate to others. An initial model is constructed showing a few of the more important features that we are interested in. The model is checked against our understanding of the situation being modelled, and additional features are slowly added to make the model more complete. Modelling is typically an iterative and evolving process, starting with very simple, basic models that are well understood but tell us little, and gradually changing through progressive refinement into more complex models that are somewhat harder to understand but tell us much. Details and features needed for better understanding are gradually added, and errors and unnecessary complexities are removed when discovered. Each stage of the model is tested against what is already known, and sometimes experimentation is used to verify implications of the model. Although models can be developed by individuals, their strength, robustness and applicability are improved if tested, verified, and validated by groups.

The entity relationship diagram

The model of most value in understanding databases is arguably the Entity Relationship model first introduced by Chen in 1976. This modelling approach has since been modified by Chen and others, and is used and understood by nearly all in the database management field. The model is principally a diagrammatic (although often augmented, in a composite manner, with text and other material) representation of the static structure of the sphere of information under consideration. Putting it more simply, the Entity Relationship Diagram is a picture showing the most important classes of things (entities) we are interested in, and how they are associated with each other (their relationship to each other), at a particular point in time. The Entity Relationship model explicitly separates data and process, and completely leaves untouched the design of the functional parts of information systems. These functional aspects of information systems are typically designed using data flow diagrams and other structured design methodologies.

Basic Entity Relationship Diagram Symbols

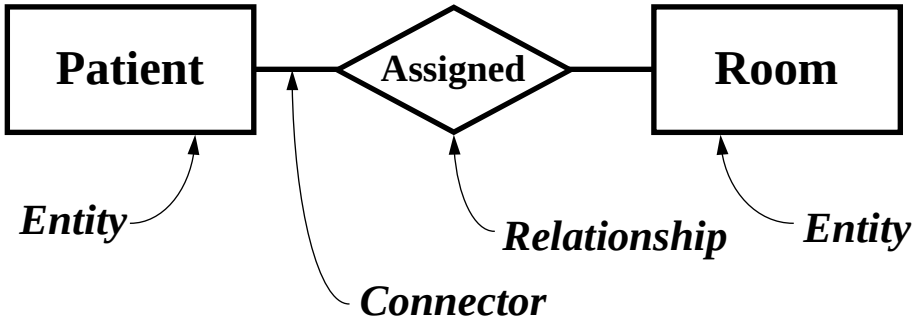

Figure 9.1 Basic entity relationship diagram symbols

In its simplest and original form, the diagram uses three symbols - a box for entities, a diamond shape for relationships, and lines connecting the entities and relationships. The box shape for entities is used to represent classes of people, places, events, and "things". "An entity may be an object with a *physical* existence—a particular person, car, house, or employee—or it may be an object with a *conceptual* existence—a company, a job, or a university course." (Elmasri & Navathe 1994 p.43, italics added) These entities are, in the real world, associated with other entities. The associations that are considered important for our model are represented by the diamond-shaped relationship symbol. In our example box we show that an entity PATIENT has a relationship ASSIGNED with an entity ROOM. Such a relationship is of interest to the hospital's admissions department, to its nursing staff, and to its food service department. We have made a satisfactory start in modelling the information requirements of our hospital.

It is likely that the different departments of the hospital are interested in different aspects and characteristics of the above entities and the relationship between them. For instance, we'd like to know the patient's name, address, next-of-kin, special dietary requirements. We would like to know exactly what room the patient is assigned, whether or not it is a private room or a large ward, whether or not it has a telephone number, and which wing of the hospital it is in. We would like to know when the patient was assigned the room, and when the patient is expected to be discharged from in-patient care. Most of these aspects and characteristics are **attributes** of the entity or relationship. Of particular interest are those attributes that identify the entity, that serve to distinguish that entity from all others. In our example, Patient-Number and Room-Number are likely to be the **identifier**s for the PATIENT entity and the ROOM entity respectively (the identifier of an entity is also known as the **primary key** of the entity). In the Entity Relationship Diagram, attributes are shown in ellipses, and identifying attributes are underlined (see the next example box).

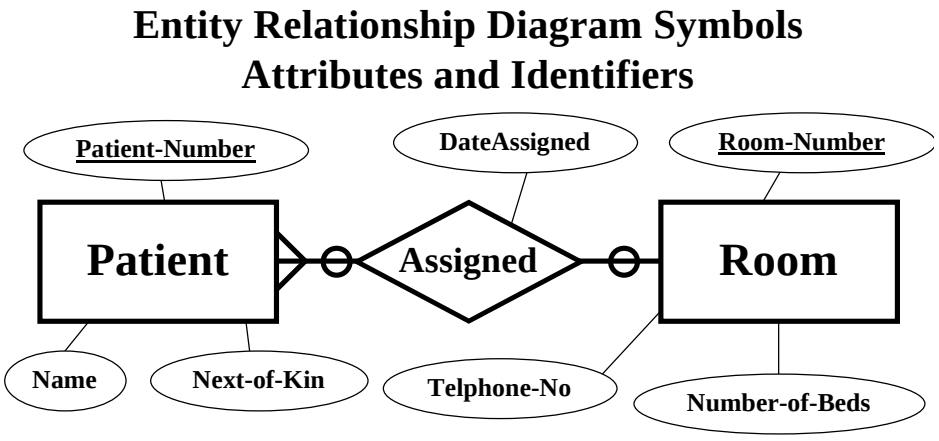

Figure 9.2 Entity relationship diagram symbols: attributes and identifiers

Also important for us to model is whether or not a patient *must* be assigned a room (out-patients may be of interest), whether or not a room *must* have a patient, whether a patient may be assigned to more than one room, and whether a room may be assigned to more than one patient. These issues are important in designing reports and forms to be used, and in estimating the space required by the database. The term used for these issues is **cardinality.** The example Entity Relationship Diagram shows that a patient is assigned to zero or one room (indicated by the circle next to ROOM), and that each room may have zero, one, or many patients assigned to it (indicated by the circle and the "crows feet" next to PATIENT).

The object-oriented data model

Early examples of object orientation can be found in the development of object-oriented-programming-languages (OOPL). Largely contemporary with the development of the Entity Relationship model, object orientation is a more comprehensive modelling approach that embodies both data and process, and is not as inherently compatible with conventional information systems as is the Entity Relationship model. For that and other reasons, although the promise of the approach has been recognised for some time, it is only recently that the object-oriented model has gained both widespread acceptance and use. However, this acceptance and use is growing at a tremendous rate, and the Object-Oriented model for information systems development may well become the standard approach in the near future.

As an approach for developing new information systems, the Object-Oriented model is a radical departure from previous practice. There is no quick and easy way to convert a traditional process designed with data flow diagrams, flow charts, and conventional programming languages to the object oriented model, and even the reverse (converting an object-oriented design into a conventional design) is difficult. The Object-Oriented approach requires a whole new way of thinking for the information systems developer. Despite these differences in modelling process, the Object-Oriented model is remarkably compatible to the Entity Relationship model, as we shall see.

Like the Entity Relationship model, the Object-Oriented model is a reflection of a view of reality. Whereas the Entity Relationship model looks at "entities" and how they are "related", the Object-Oriented model looks at "objects" and includes in this analysis behaviour as well

as attributes. From this perspective, the Object-Oriented model can be viewed as an extension of the Entity Relationship model.

Central to the Object-Oriented model is the concept of *encapsulation*. Encapsulation is a "packaging" of the various characteristics of objects, both descriptive and behavioural. The object has public, or external, characteristics that can be seen and affected by the outside world by means of *messages* sent to it. It also has private, or internal, characteristics that embody the implementation mechanisms that are not needed by the outside world and cannot be directly affected by it. The descriptive characteristics in the Object-Oriented model are called the *attributes* of the object, with the word having much the same meaning and use here as in the Entity Relationship model. The behavioural characteristics are modelled by means of *methods*, with the only means of accessing and modifying the attributes being via the object's methods, often requested by other objects by way of a message.

One major contribution of the Object-Oriented model is its explicitly formalised use of generalisation. Object *instances*, or occurrences, are members of a *class* of similar objects, with similar attributes and methods. Object classes can in turn be grouped into more general classes, or subdivided into subclasses. A subclass will *inherit* most of the attributes and methods of its more general class, and will introduce other attributes and methods specific to it.

Objects can be assembled into more complex objects, complex objects may be subdivided into more simple ones. One important characteristic of object classes is that each instance of an object is distinct and distinguishable from every other object instance, and this differentiation is maintained no matter what changes are made to its attributes. Each object instance thus has its own *identity*.

The relational data model

Both the Entity Relationship model and Object-Oriented model above are useful in the earlier stages of design for a database. Both models express some high level ideas that are important for getting the design done correctly. However, neither model can be directly implemented. With the Entity Relationship model we need a conventional database management system, with the Object-Oriented model we need an Object-Oriented Database Management System or at least an Object-Oriented Programming environment. The relational data model may be used to attach the extra detail needed to the Entity Relationship model to allow the implementation using a relational database management system.

Tables of rows and columns

Conceptually, the relational model consists of nothing more than tables of rows and columns, and operations on those tables. The relational model would represent the Entity Relationship diagram above using the following tables:

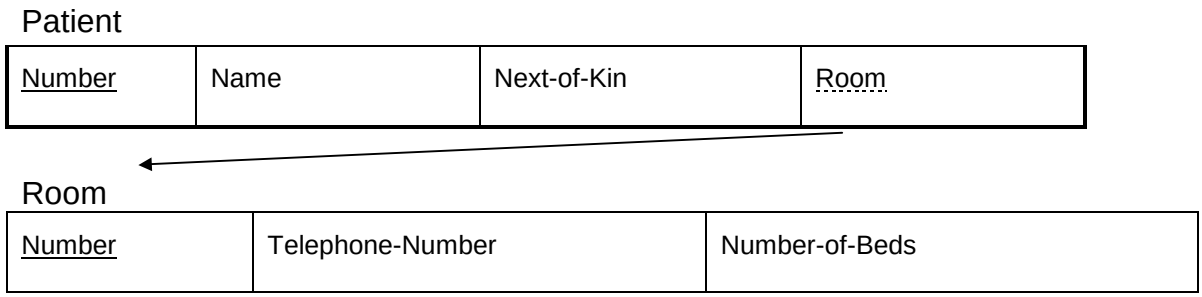

Figure 9.3 Relational data model

Note that the relationship "assigned" is modelled by including a reference to the room in the patient table (called a *foreign key*).

Information resource management

Information Resource Management is a concept that first became popular in the middle 1980's. At the time there was a growing dissatisfaction by the senior management of many organisations with the performance of their Information Systems departments. These departments were seen to be costing more and more, yet were consistently falling behind on delivering what the organisation needed. A few organisations were spectacularly successful in using their information systems investment for strategic advantage in competitive markets, more organisations were still waiting for developments promised as much as three years previously. The information systems professionals responded to the criticism by noting that the development of appropriate information systems was a critical factor in implementing the organisation's strategies. Prior to this period, information systems development tended to operate in a reactive manner, with the squeaky wheel being greased. Planning was often limited to guessing what new hardware would be required, with little thought to what development projects should be undertaken. Information Resource Management starts with the premise that the information used by the organisation is of strategic value, and should be managed with the same care and attention as other corporate resources. Corresponding to the important role played by an organisation's Chief Financial Officer, the zealots of information resource management suggest that the function of Chief Information Officer be created and vested with similar powers and responsibilities.

The strategic role of databases

By taking a broad view of the organisation's information resources, future uses of information technology can be anticipated and accommodated without needing to replace or discard the existing investment in information systems. It has been observed that the data used by an organisation is relatively stable, whereas the processes that use this data, and the structure of the very organisation, is not. Separating the design of databases from that of the organisational units and structures that use the data gives a high degree of independence, stability, and value. Databases that have been designed with modern, well-developed concepts can be used to lever the effect of new technology and innovation, and not hold these effects back.

Planning for databases

"The goal of the planning phase is to align information technology with the business strategies of an organisation" (McFadden & Hoffer 1994 p.81) Even with workbench technologies, prototyping, and other rapid information systems development approaches, it still can take a substantial amount of time, effort, and expense to create the databases needed to fulfil the strategic needs of the organisation. Organisations cannot afford to spend these resources unless by doing so they are furthering the goals of the organisation as a whole. Organisations that can respond quickly to changes in their environment will outperform those that cannot. How can expenditures be managed and quick responses facilitated? By having a plan for the future, and working to it. The planning for databases is a fundamental part of strategic information systems planning, and needs to be closely integrated with the strategic planning of the entire organisation. Databases which support the critical success factors of the organisation must receive the highest priority.

Planning for databases is a process involving modelling the organisation, identifying and prioritising the information needs of the organisation, and developing a plan to progressively develop and improve the provision of critical information to the decision makers of the organisation. The plan itself is flexible, with regular review and provision for modification. Distant targets are provisionally sketched in, with detail increasing as the time horizon shrinks.

Distributed databases

Most organisations that might be interested in database technology are dispersed to a fair degree, occupying at least several buildings, and often having offices separated by hundreds of kilometres. To such organisations, the ability to share information quickly and easily across the organisation is important or essential. Mail systems, couriers, telephones and facsimile machines are all used to enable this dissemination. With computer networks, the computer-based information resources of the organisation can be easily on tap wherever needed.

Approaches to distribution

McFadden & Hoffer (1994 p.473) list four basic strategies for distributing databases.

1 data replication, where some or all of the database is copied to different locations;
2 horizontal partitioning, where the database is segmented according to the value of one or more of each record's attributes (for example, patients listed with a Rockhampton postal code would have their patient data stored in Rockhampton, those with a Brisbane postal code would have it stored in Brisbane);
3 vertical partitioning, where the database is segmented according to the different functions using it (for example, a hospital's detailed budget information may be kept in computers within the hospital, but the higher level budget information may be kept in computers within the regional offices);
4 combinations of replication, horizontal and vertical partitioning, (for example, where tables of valid pharmaceutical items may be copied and kept in all hospitals, pharmacy inventory kept by each pharmacy, and drug use statistics maintained monthly in Brisbane).

Reasons for distribution

A database may have a number of reasons why it should be distributed. These reasons are usually based on cost, performance, or both. Compared with a centralised database, distributed systems offer the potential for greatly reduced communication charges, since the data resides closer to where it is actually used. Control of costs can be devolved, and local autonomy and accountability increased, with distributed databases. Response times for the more common transactions should improve considerably, since the bottleneck of the central computer has been removed. Parts of the network may be able to keep operating when the central site or network is malfunctioning. Some sites may be able to operate at different hours from others, they may not be constrained by the needs of others. Growth may be accommodated gradually, without the need for drastic spurts.

Implications of distribution

Whatever the approach used for distribution, the software and hardware required will likely be more complex than for a centralised system. This will make it more expensive, and could cause it to have more problems. Queries that span the data from several sites may be considerably slower. Data replication implies a greater risk for database integrity, inadequately implemented replication could both increase transmission costs and reduce the security and integrity of the database. Devolution without adequate resourcing can place a significant burden on the operating departments, without yielding the benefits promised.

Database security and integrity

Standard features with larger database packages are systematic and largely automated *backup* and *recovery* mechanisms. These features should be implemented to ensure that the database is adequately protected from the failure of either hardware, software, or the network. The power and value of databases comes from being able to share the data. *Concurrent access* by many users must be enabled in order to maximise this value. Some database management systems that grew from personal computer programs may not have adequate performance and protection in this area.

Data kept in a medical database is often confidential. Damage to this data may threaten lives, whether the damage is intentional or accidental. A characteristic of good database management systems is the ability to identify who should have access to what parts of the database, and *restrict access* to the remainder of the database or to unauthorised people. Of particular concern is data distributed over a network.

We have argued that the organisation's database is a valuable, strategic resource. Plans should be in place, and exercised, to avoid the worst damages that might be caused by a disaster.

References

Chen, P. (1976) "The Entity Relationship Model - Toward a Unified View of Data", *Transactions on Database Systems*, 1:1, March 1976

Elmasri, R. and Navathe, S.B. 1994 *Fundamentals of Database Systems*, Benjamin/Cummings, Redwood City, California, page 43.

McFadden F.R., and Hoffer, J.A. (1994), *Modern Database Management*, Benjamin/Cummings, Redwood City, California