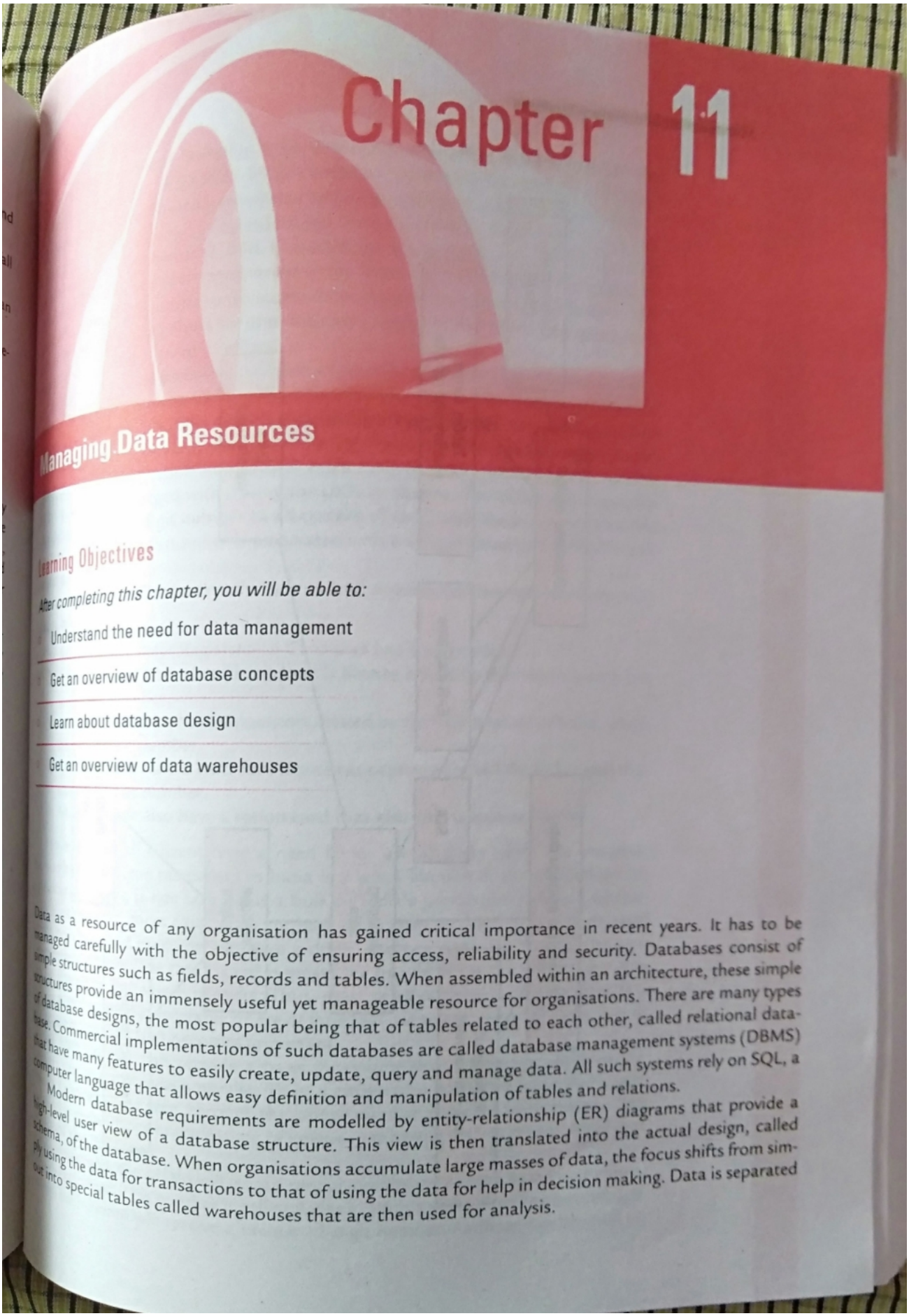


Chapter 11

Managing Data Resources

Learning Objectives

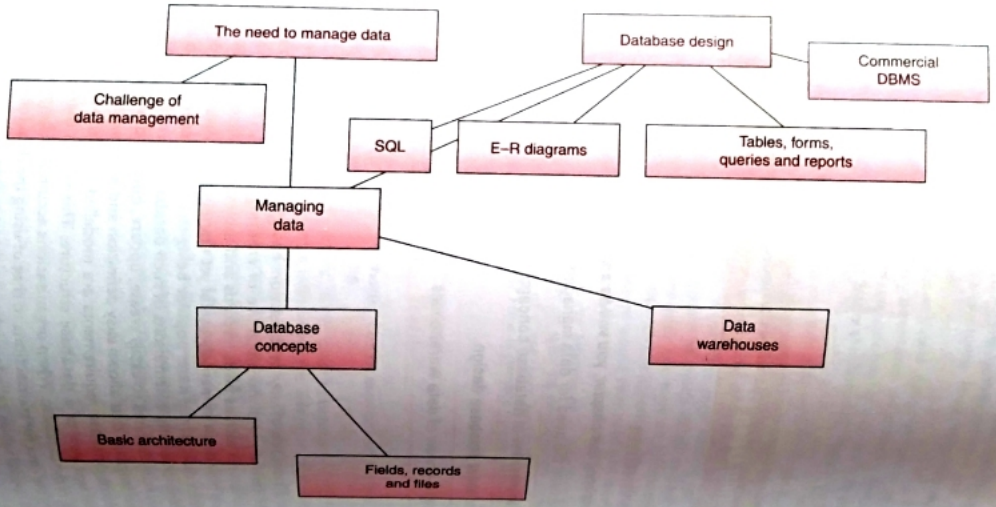
After completing this chapter, you will be able to:

- Understand the need for data management
- Get an overview of database concepts
- Learn about database design
- Get an overview of data warehouses

Data as a resource of any organisation has gained critical importance in recent years. It has to be managed carefully with the objective of ensuring access, reliability and security. Databases consist of simple structures such as fields, records and tables. When assembled within an architecture, these simple structures provide an immensely useful yet manageable resource for organisations. There are many types of database designs, the most popular being that of tables related to each other, called relational database. Commercial implementations of such databases are called database management systems (DBMS) that have many features to easily create, update, query and manage data. All such systems rely on SQL, a computer language that allows easy definition and manipulation of tables and relations.

Modern database requirements are modelled by entity-relationship (ER) diagrams that provide a high-level user view of a database structure. This view is then translated into the actual design, called schema, of the database. When organisations accumulate large masses of data, the focus shifts from simply using the data for transactions to that of using the data for help in decision making. Data is separated out into special tables called warehouses that are then used for analysis.

MIND MAP OF CHAPTER TOPICS



CASE STUDY: Unique Identification Number in India

For citizens of India, mobility from one state to another is a problem. If one moves, say, from Uttar Pradesh to Karnataka, then in the new state of residence, one will have to obtain a new driver's licence, open a new bank account, obtain a new permit for cooking gas cylinders, get a new electricity connection, re-register an old vehicle in the new state, and, if needed, get a new ration card. This is because these documents cannot be transferred easily from Uttar Pradesh to Karnataka, as there are no legal provisions to do so. As these documents require considerable time and effort while getting them for the first time, applying and waiting to get them a second time is a huge waste of effort.

It is partially to address this problem of transfer of documents that the Government of India initiated the Unique Identification Number (UID) scheme. Under this scheme, every citizen of India will be provided a unique number that will be backed by an infrastructure to verify and authenticate the number. A special organisation, called Unique Identification Authority of India (UIDAI), was created in 2009 for this purpose, and was charged with issuing the UIDs to citizens. The UIDAI will eventually provide a unique 12-digit number to all citizens of India with the assurance that the number is unique (each number is associated with a unique citizen), is verifiable and is valid across India.

Many citizens of India already have several documents that provide them with unique numbers:

1. Most have a Voter Registration Card that has a unique ID.
2. Many citizens have an official driver's licence issued by their state, which has a unique number.
3. Many citizens also have a passport, issued by the Government of India, which has a unique number.
4. Many citizens have a card for income tax payment (called the PAN card) that uses a unique number.
5. Many citizens also have a ration card that also uses a unique number.

However, not all citizens have a need for or use all these cards. For instance, the number of income tax payers in India is a small fraction of the population (as agricultural income is not taxed and a bulk of India's population relies on agriculture). Furthermore, most citizens do not have a passport, as they don't need to travel across borders, and many do not have a driver's licence either, as they do not own a motorised vehicle. The ration card is meant for people below the poverty line, but can be issued to any citizen of India. Thus, most or all of these cards that provide a unique number are not available or are of little real use to all citizens of India. It is in this context that the UID becomes important.

An UID number can provide a basis for uniting these disparate identification projects under a common umbrella. Thus, a citizen who has a PAN card and also a driver's licence can be seen, through the unique number, to be the same person. This will reduce redundancy in the issuing of unique numbers as well as control fraud and misuse of the numbers.

As envisaged, the UIDAI will issue a unique number to citizens based on biometric authentication. This number is called *Aadhaar*. With a scan of ten fingerprints and the iris, each citizen will receive a unique 12-digit number. Aadhaar can be used by

CASE STUDY: Unique Identification Number in India

banks, or the tax authorities, or schools, or the ration card agencies to issue cards or validation documents to citizens. The idea is that if a citizen presents a bank card to a merchant for some commercial transaction, the merchant can verify that the card belongs to the particular individual by checking against the UIDAI database. This verification service will be made available at a reasonable cost by the UIDAI.

Aadhaar, in this sense, becomes what in database terminology is called a *primary key*, a unique identifier for a record that can be used across the database and across applications without worry of duplication. Various agencies like banks, the tax authority, the passport agency, the motor vehicles department and the public distribution system can then use Aadhaar to issue their own verification and authentication documents. A citizen can move from one part of the country to another, and with Aadhaar he/she can retrieve or use his/her card anywhere and be assured that his/her identity is authenticated.

The role envisaged for Aadhaar is best captured by the Chairman of the UIDAI, Mr. Nandan Nilekani, 'The name Aadhaar communicates the fundamental role of the number issued by the UIDAI the number as a universal identity infrastructure, a foundation over which public and private agencies can build services and applications that benefit residents across India'.

1. Aadhaar's guarantee of uniqueness and centralised, online identity verification would be the basis for building these multiple services and applications, and facilitating greater connectivity to markets.
2. Aadhaar would also give any resident the ability to access these services and resources, anytime, anywhere in the country.
3. Aadhaar can, for example, provide the identity infrastructure for ensuring financial inclusion across the country – banks can link the unique number to a bank account for every resident, and use the online identity authentication to allow residents to access the account from anywhere in the country.
4. Aadhaar would also be a foundation for the effective enforcement of individual rights. A clear registration and recognition of the individual's identity with the state is necessary to implement their rights – to employment, education, food, etc. The number, by ensuring such registration and recognition of individuals, would help the state deliver these rights.

Source: uidai.gov.in (accessed on June 2011).

Aadhaar in India is similar to a unique number given to citizens in other countries. In the USA, all citizens are required to have a Social Security Number (SSN) that was originally designed to provide them with social security – such as pension, medical care, job-loss compensation and so on – but it is now used for many different purposes such as for opening a bank account, obtaining a driver's licence, getting a credit card, being registered for medical insurance, enrolling in school or college and so on. Such schemes for providing social security, along with a unique number, are prevalent in European countries too, such as Spain and France. In all these countries,

the unique number is used for many transactions other than social security, including those of credit card purchases, property purchases and college enrolment.

The Aadhaar scheme has come in for a fair measure of criticism. For a country as diverse and complex as India, critics argues, such a scheme is not suitable. Some argue that the Aadhaar scheme will link many vital sources of information about individuals under a common source and thus compromise individual privacy. Those with dubious intentions can snoop into online and computerised records of individuals and have access to a vast store of information, something that is not possible without a primary key. Others contend that, in the case of poor and marginal citizens, obtaining and maintaining such a number will become an additional burden, and instead of helping them, it will further impede their ability to make a living and function effectively. Still others argue that Aadhaar will become another tool in the hands of a corrupt and power-hungry bureaucracy, which will extract further rents from those unable to understand the value of this scheme and how it can be used effectively.

11.1 THE NEED FOR DATA MANAGEMENT

11.1.1 History of Data Use

In the early years of computing when programs were written on large mainframe computers, the practice was to include the data required for a computation within the program. For example, if a program computed the income tax deductions for personnel in an organisation, the data regarding the personnel and their income was maintained within the program itself. If changes were required, say, when a new employee joined the organisation, the entire program for income tax calculations would have to be modified, not just the data alone. Changes to data were difficult as the entire program had to be changed, and further, the data was not available to other programs.

With advances in programming languages, this situation changed and data was maintained in separate files that different programs could use. This improved the ability to share data, but it introduced problems of data updating and integrity. If one program changed the data, other programs had to be informed of this development and their logic had to be altered accordingly.

A start in organising data came with the idea of the relational data storage model, put forward by British scientist E.F. Codd in 1970. Codd, then working with IBM in the USA, showed how data could be stored in structured form in files that were linked to each other, and could be used by many programs with simple rules of modification. This idea was taken up by commercial database software, like Oracle, and became the standard for data storage and use.

11.2 CHALLENGE OF DATA MANAGEMENT

Consider the following facts:

1. Researchers estimate that the total amount of data stored in the world is of the order of 295 exabytes or 295 billion gigabytes. This estimate is based on an assessment of analog and digital storage technologies from 1986 to 2007. The report (by M. Hilbert and P. Lopez appeared in *Science Express* in February 2011) states that paper-based storage of data, which was about 33% in 1986, had shrunk to only about 0.07% in 2007, as now most of the data is stored in digital form. Data is mostly stored on computer hard disks or on optical storage devices.
2. The consulting firm IDC estimated (in 2008) that the annual growth in data takes place in two forms:
   (a) *Structured:* Here data is created and maintained in databases and follows a certain data model (explained in Section 11.4.4). The growth in structured data is about 22% annually (compounded)
   (b) *Unstructured:* Here data remains in an informal manner. The growth in unstructured data is about 62% annually.
3. The large online auction firm eBay has a data warehouse of more than 6 petabytes (6 million gigabytes), and adds about 150 billion rows per day to its database tables.

The above examples highlight the incredible amounts of data that are being created and stored around the world. Managing this data so that it could be used effectively presents a strong challenge to database systems: the systems not only have to store the data but also have to make it available almost instantly whenever needed, allow users to search through the data efficiently, and also ensure that the data is safe and uncorrupted. Different aspects of the need for database systems are discussed in the sections given below.

11.2.1 Data Independence

Databases allow data pertaining to an activity or a domain to be maintained independently. This independence means that the data is stored in separate files in a structured manner, and the creation and updating of the data is done independent of its uses. For instance, in a college, a database of students is updated when a student joins or leaves the college, changes address, changes phone number and so on. This is independent of how the data is used by programs for course registration or for the library. Furthermore, the programs and applications that use the data are not aware of where and how the data is maintained; they only need to know how to make simple calls to access the data.

11.2.2 Reduced Data Redundancy

One goal of databases is to reduce data redundancy. Data redundancy refers to the duplication of data in different tables. If data on students is maintained in two or three different databases in the college then for one change, say in a student's mobile phone number, all the databases have to be changed. Reduced data redundancy ensures that minimal storage is used for the data. With the rapid increase in data over time, conserving space is an important management challenge.

11.2.3 Data Consistency

It is important that data users have access to consistent data, that is, the data is the same regardless of the application through which the user accesses it. Consistency implies that the integrity of the data is maintained (the data has not been changed or corrupted in a manner unknown to the system); the data is valid, which means that the data is the correct one to use for the moment; and the data is accurate, which means that the data being used is the one that was designed to be used. Consistency requires careful management of data updating, deletion, copying and security.

11.2.4 Data Access

Data stored in databases must be accessible efficiently. Very large databases, such as those maintained by eBay, have to be managed in a way that when users search within them, their results should be available within a matter of seconds. A search in eBay results in a response within a few seconds, even though the system has to search

through billions of records. Furthermore, the response from the database has to be presented to the user in a manner that is easy to read and understand, which requires further processing.

11.2.5 Data Administration

Data administration entails deciding who can *create*, *read*, *update* or *delete* data. Many organisations have strict controls over who can create or delete data fields or tables. This is determined by the needs of the organisation and the roles defined for database administrators and users. Read access is usually provided to those who need to only see and use the data, but not modify or change it in any way. Update access is also carefully restricted to those who have the rights and privileges to do so. Modern database systems enable sophisticated ways in which these four functions can be enabled or disabled for users and administrators.

11.2.6 Managing Concurrency

A serious challenge for modern databases, especially those used for e-commerce applications, is that of managing concurrency. Data is often maintained on many servers, distributed across a wide geography. Concurrency entails ensuring that changes or updates to a particular element in a table are reflected across all the distributed servers where users access the data. This is an element of managing consistency, particularly for distributed databases.

11.2.7 Managing Security

A substantial threat to modern databases is from crackers and unauthorised users. Database systems have to provide a layer of security, over and above the security systems in place at the organisation, which ensures protection across transactions and all administration tasks. This also means that internal tampering and mischief with data is carefully monitored and controlled.

11.2.8 Recovery from Crashes

Databases are crucial to the internal working of an organisation – they are both a resource and an asset. With the high levels of transactions happening within the IS of organisations, it is imperative that the data is secured against failure. Modern database systems provide a sophisticated system of backup, mirroring and recovery that allows rapid recovery from crashed servers.

11.2.9 Application Development

Databases enable applications to be developed using the facilities of data management provided by them. E-commerce sites, for example, create a web presence that includes search, display, selection, sale and payment for products, which rely on databases that provide all the relevant data, and store data, for the transactions.

Applications may be local to a function or department or shared across many departments, and they may share data from the databases. Database systems provide special languages by which their data can be manipulated, and hence can be used by application developers.

11.3 DATABASE CONCEPTS

A database is a collection of files that have stored data. The files and the data within them are related in some manner – either they are from the same domain, the same function, the same firm or some other category. The files in the database are created according to the needs of the function or department and are maintained with data that is relevant for the applications the department runs.

An example of a database in an organisation is an 'Employee' database. This will correspond to the human resources function of the organisation. The 'Employee' database may contain files related to employee details, their employment history, their benefits details, their leave history, their family details, their medical compensation details and so on. The files are related to the employee concept, although they contain different data, depending on the applications that will need the data. Computations regarding medical reimbursements, for instance, will read data from the files related to the employee's status, benefits and medical history.

Consider another example of a 'Product' database. This may contain files related to the types of products, the details about product features, the prices and history of prices of products, the regional and seasonal sales figures of products and the details of product designs. Such files could be used by the manufacturing department to determine production schedules, by the marketing department to determine sales campaigns or by the finance department to determine overhead allocations.

11.3.1 Fields, Records and Files

A file in a database consists of particular spaces, called structures, in which data is maintained. The basic structure of a file is called a *field*. A field is a defined space, of specific dimensions, in which data is placed. Data is read from and can be deleted from fields. When defining or designing a field, the contents of the field have to be specified exactly. For instance, if the field has to hold the date of birth of an employee, it has to be specified how the data will be stored (in dd-mm-yyyy format or mm-dd-yy format), and what kind of characters (numbers, in this case) will be permitted. There are several other dimensions that specify a field and are held in a *metadata* file. (A metadata file contains details about how data is stored in files, and provides information on how the data can be used and managed.)

A collection of fields is called a record. Each record is like a row in a spreadsheet; it consists of a pre-defined number of fields. In most cases, the sizes of the fields are fixed; this ensures that the total size of a record is also fixed. Records are a collection of fields that have meaning in the context of the application. For example, consider the two records of a student file given in Table 11.1.

Table 11.1 Two Records of a Student File

Aadhaar Number	First Name	Last Name	Year of Birth	Major
234577643239	Aamir	Khan	1968	Physics
–	–	–	–	–

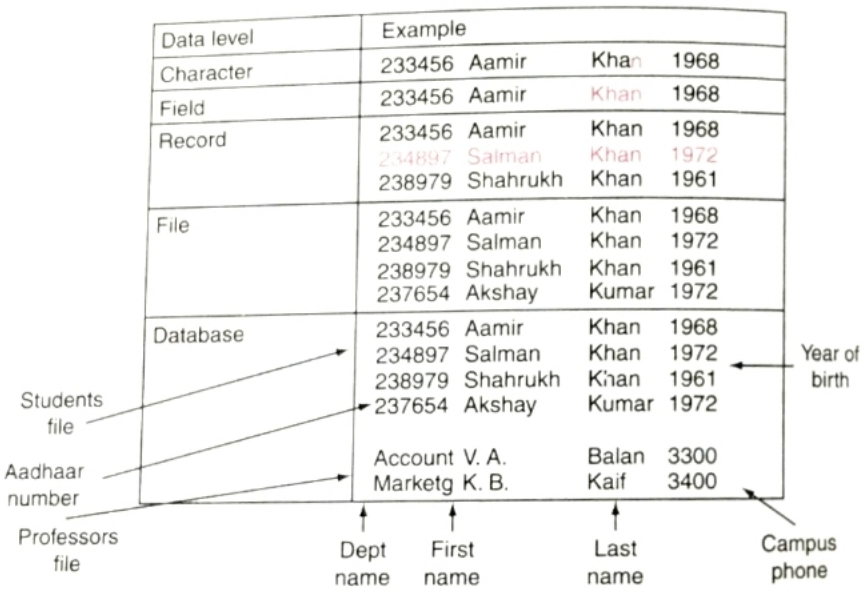


FIGURE 11.1 The basic elements of a database – fields; records and files.

Table 11.1 shows five fields that define a record. Each field contains data pertaining to some aspect of a student - roll number (Aadhaar number in this case), first name, last name, year of birth and the subject in which the student is majoring. The data in each field is written in characters or numbers. For each record, there should be at least one field that uniquely identifies the record. This ensures that even if there are two students with exactly the same name (say Aamir Khan), with the same year of birth and the same major, then there is at least one field that will distinguish the records of the two students. In Table 11.1, Aadhaar number is the unique identifier. In other cases this could be an employee number, a tax number or even a random number generated by the system. This unique field is called a *primary key*.

A table is contained in a file (see Fig. 11.1). Each table may contain a few records or a very large number of records. A database consists of many files. Modern database systems allow table sizes to include billions of records. Furthermore, very large tables may be split and stored on different servers.

In relational databases, the tables are related to each other. These relations allow data to be linked according to some logic and then extracted from the tables. A detailed example of this is provided in a later section.

11.3.2 Basic Architecture

Databases may be organised and used in many different ways. The most basic use is as a *personal database*. Individual users create databases for their personal use in organisations or at home. A personal database may be on a personal computer at office, on a mobile phone or on a tablet computer. The data in these databases is fed and updated by the user, and is principally used by him/her. For instance, a contacts database on a mobile phone is a personal database. All the data is entered and used by the mobile phone user. The design of the database is not created by the user (such databases are often provided as off-the-shelf applications), but the use and maintenance is only by the user.

Personal databases are highly tuned to the needs of the user. They are not meant to be shared. These databases also cannot be shared, as they reside on personal devices; and this is a limitation of these systems.

Workgroup databases or function databases are designed to be shared across employees in an organisation, either belonging to a group or to a functional department. Such a database is maintained on a central computer, along with applications relevant for the group or department. Users access and update the data on the central database from their local computers.

Enterprise or organisational databases are accessed by all members of the organisation. These are typically organised in the client–server mode (see Fig. 11.2). A central database server provides database capabilities to different applications that reside on other computers. These client applications interact with the database server to draw on data services, whereas the database server is managed independently. An advantage of these database servers is that they can be made highly secure, with strong access restrictions, and can also be backed up carefully to recover from crashes.

While designing client–server databases, a prime issue to be addressed is – where the processing should take place. If data processing has to be done on the client from, say, three tables then these tables have to be moved across the network to the client, which should have enough computing capacity to do the processing. If, on the

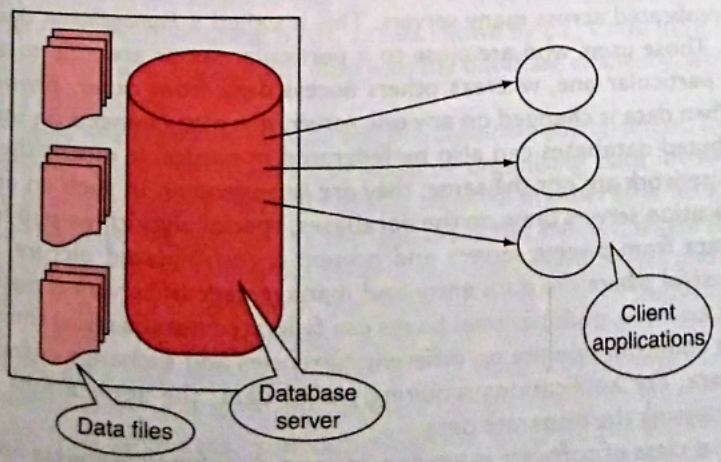


FIGURE 11.2 Client–server architecture of a database.

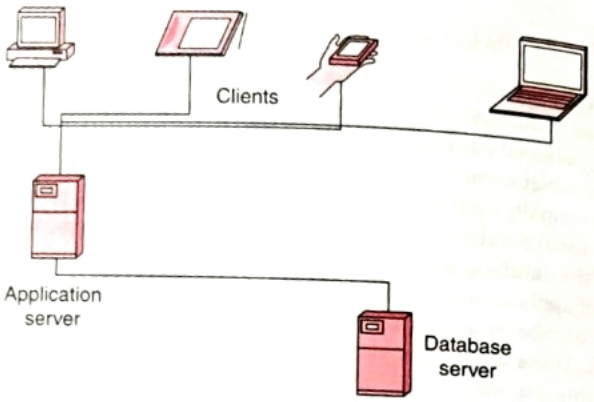


FIGURE 11.3 Three-tier database architecture.

other hand, the computing is done on the server then the clients have to send processing requests to the server and await the results, and this puts a lot of load on the server. Clients such as mobile phones or personal computers often do not have the processing capacity to deal with large amounts of data, so the processing is invariably left to the server.

The architecture often used in enterprises is referred to as three-tier architecture (see Fig. 11.3). Here the clients interact with application servers, which then call upon database servers for their data needs. Here the load of processing for applications and for data is spread across two sets of servers, thus enabling greater efficiency.

Databases may be centralised or decentralised within organisations. Centralised databases are designed on the client–server model, with a two-tier or three-tier architecture. Decentralised or distributed databases have tables distributed across many servers on a network. The servers may be geographically distributed, but for the applications they appear as a single entity. One type of distributed server has the entire database replicated across many servers. This is called a *homogeneous* database (see Fig. 11.4). Those users who are close to a particular server are able to access data from that particular one, whereas others access data from other, physically closer servers. When data is changed on any one server, it is also changed on the others.

Distributed databases can also be federated in nature. It means the databases across the network are not the same; they are *heterogeneous*. In such an architecture, when application servers draw on the databases, special algorithms pull together the required data from diverse servers and present a consolidated picture. This architecture is useful where the data entry and management of servers is particular to a region. For example, multinational banks use federated databases as their databases in different countries operate on different currencies and exchange criteria, and rely on local data. For applications requiring global data, the applications use special logic for analysing the disparate data.

A special class of software is used to connect disparate databases and these are known as *middleware*. As databases can have different data structures for the same kind of data, the middleware software allows the databases to read and write data

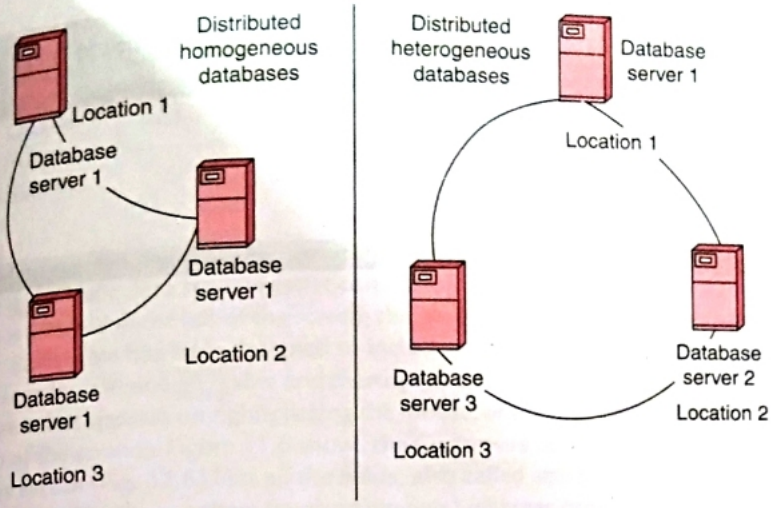


FIGURE 11.4 Distributed databases with homogeneous and heterogeneous architectures.

to and from each other. For example, the data field for 'student name' may have a space for 30 characters in one database and 40 characters in another. The fact that they are referring to the same concept is captured by the middleware that enables the translation from one to the other. The middleware is also used by the Application Layer to read and use data from many databases. In modern web-centric applications, the middleware plays a major role in allowing the use of distributed databases by application servers.

11.4 DATABASE DESIGN

Among different types of databases, relational databases are the most widely used. Relational databases consist of tables, or files, linked together by 'relations' that are specified according to the needs of the application that will use the data. Such a relational design has a mathematical basis, as was shown by E.F. Codd in 1970, and can be manipulated to extract and input data into the database efficiently, while ensuring integrity and consistency.

Other types of databases include the hierarchical, object-orient, network and object-relational types. The *hierarchical model* of organising data involves creating a structure in which data is maintained in a tree-like manner (the 'tree' is actually an inverted tree with the root at the top and branches below), where the topmost structure is the root from which the lower structures inherit values and properties. This model is not widely used now, but there do exist certain applications where it is highly suitable. The *object-oriented model* relies on structures that are objects, which include data and procedures that act on data. Object-oriented databases are designed to work well with object-orient programming languages, which are popular high-level languages, and can efficiently integrate the database with the application. The *network model* places data objects and relations in a graph-like structure. This model too is used in special applications. The *object-relational model*

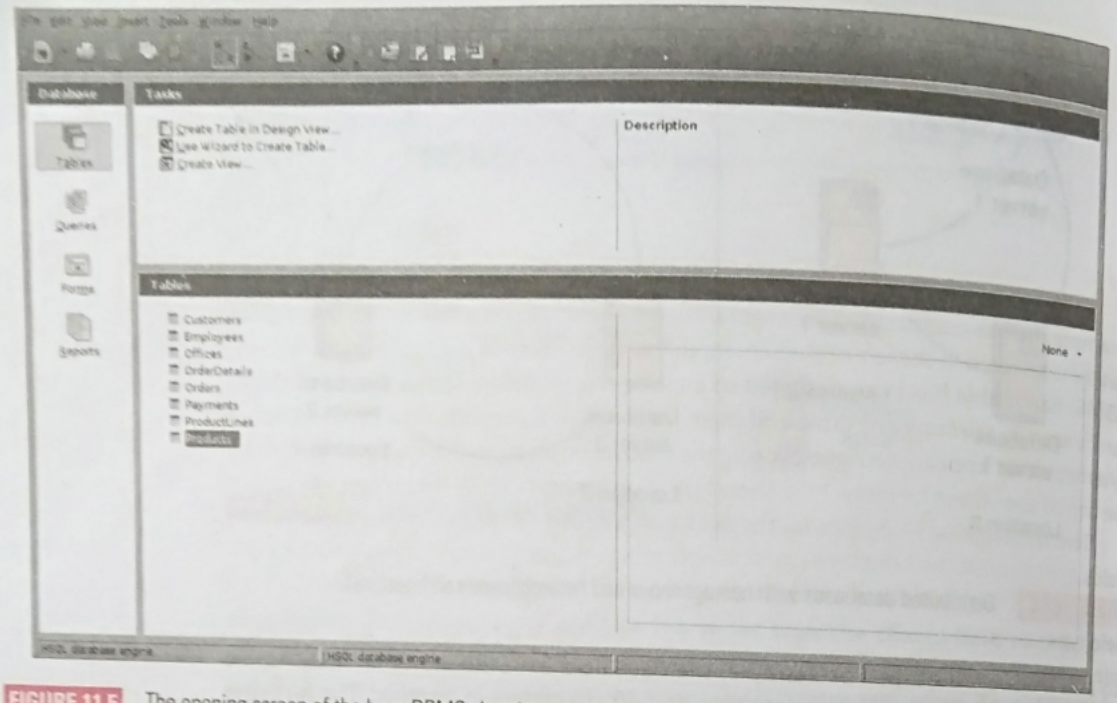


FIGURE 11.5 The opening screen of the base DBMS showing the 'Classic Models' database.

enables the use of objects in relational tables. Data is maintained in objects, and these are related to allow reading and updating of the data with specified procedures. Many modern database systems support this model, and it is gaining popularity.

Although there are several competing models, the relational database model remains the most popular. The rest of this section focuses on the design of relational databases.

The first design issue for a relational database is that of designing the tables that will hold the data. Tables consist of records, which are constituted by fields. The manner in which tables are constructed depends on the DBMS being used. Most modern DBMS allow tables to be created using a visual interface, with tool bars and drop-down menus. However, tables can be created by using the SQL language also. Figure 11.5 shows the opening window of an open source DBMS called Base (which is part of the Open Office suite).

Figure 11.5 shows a sample database called 'Classic Models'. The database is for a firm that makes and sells toy models of classic cars, planes and ships. The data is about customers, products, employees, payments, offices and orders. (This database is freely distributed under the Eclipse Public License, as part of the Eclipse Birt project. Details can be found at www.eclipse.org/birt)

11.4.1 Elements of Databases

Using the Classig Models database, we can now understand how the elements of a database can be created in a DBMS. The following sections show how tables, forms, queries and reports are created in the DBMS.

As is seen in Fig. 11.5, the DBMS has four categories of objects in the left panel:

1. Tables.
2. Queries.
3. Forms.
4. Reports.

11.4.1.1 Tables

A table is the basic data structure that consists of fields and records. When the 'Tables' feature is selected in the left of the screen, the system shows all the tables already in the DBMS. Each table has been designed to include certain fields. The design of the table can be seen by selecting any table and then opening it in Edit mode (this can be reached by a menu that appears on right-clicking the mouse, or through the 'Edit' menu item on the top of the screen). Figure 11.6 shows the Customers table in the design view.

The screen (Fig. 11.6) lists all the fields, also called attributes, of the table. Some fields are specified as numbers (such as integers) whereas others are specified as text characters. The first field is called the `customerNumber` and it is declared to be a field type of 'integer'. This implies that the `customerNumber` field will contain only integer-valued numbers. If any other type of data is placed in the field (say some text values), this will constitute an error, and the DBMS will not allow it to be stored. In the lower half of the screen is a window in which the properties of the field can be provided. For instance, the length of the field, its default value and how the data has to be formatted are specified. The properties also specify whether a value for this field has to be supplied or it can be left blank.

The properties of all the fields are similarly specified. The system treats each field differently. For example, for mathematical calculations, only number-based field types will be used. A field containing data about, say, prices will be a numeric field, whose values can be added, subtracted, etc.

11.4.1.2 Forms

Once a table is designed, it can be populated with data. One way to write data into tables is by using a *form*. A form is a user interface of a table (or set of tables) that allows specific users to enter or change data in tables (Fig. 11.7). Forms can be designed using the graphical interface of a DBMS. Forms are always built on top of a table or several tables. Once the Forms item is selected in the Base screen, it shows that a form can be created by using a wizard or it can be designed directly. A form allows a particular user, or set of users, to update data regarding some aspects of a table, and nothing else. For example, in the Customers table, one form (Fig. 11.8) can allow some users to only change the address and phone numbers of the customer and nothing else.

11.4.1.3 Queries

As a form enables entry of data into tables, a query allows reading of data from tables. At its most basic level, a query allows selection of data from a table according to some criteria. For example, the Customers data table can be queried to extract the names, phone numbers and credit limit of customers from Finland. The design for this query can be done with a wizard in Base, as shown in Fig. 11.9. Figure 11.10 shows the table that results from running the query on the Customers table.

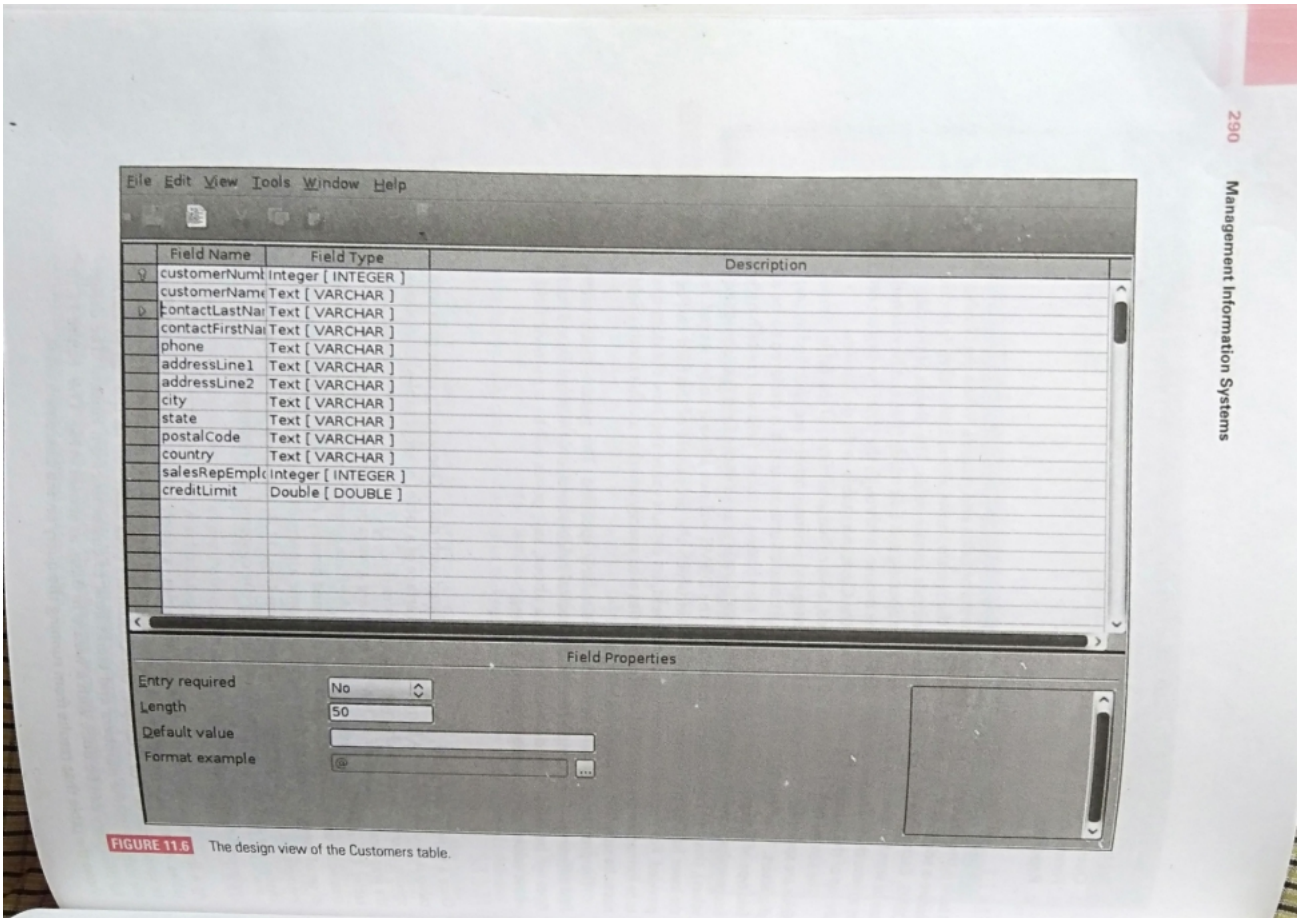


FIGURE 11.6 The design view of the Customers table.

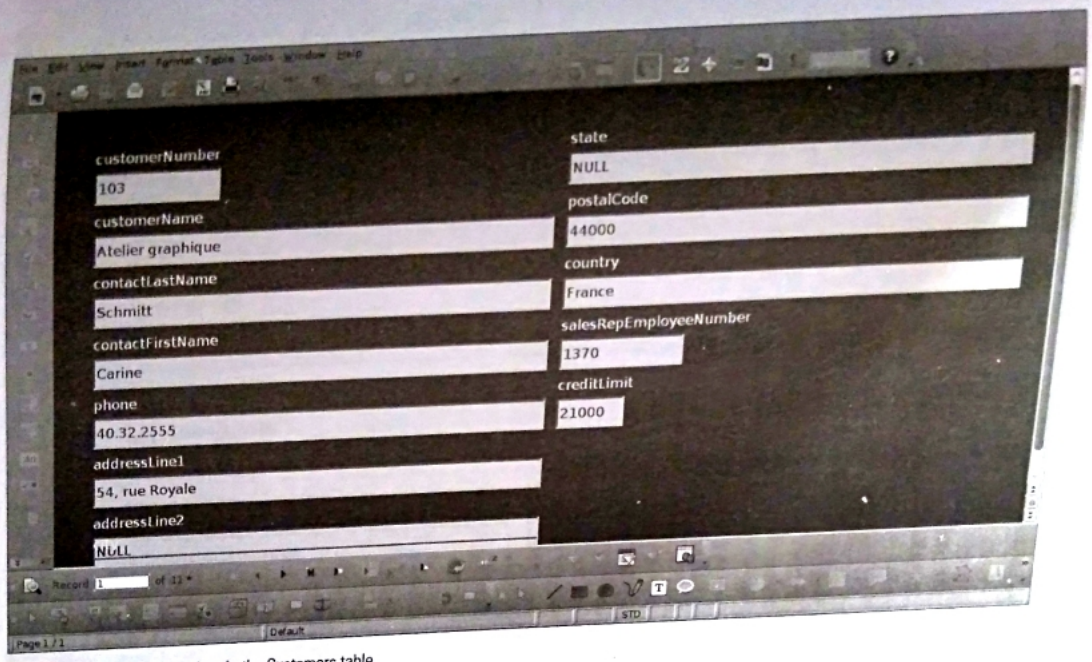


FIGURE 11.7 Form for entering data in the Customers table.

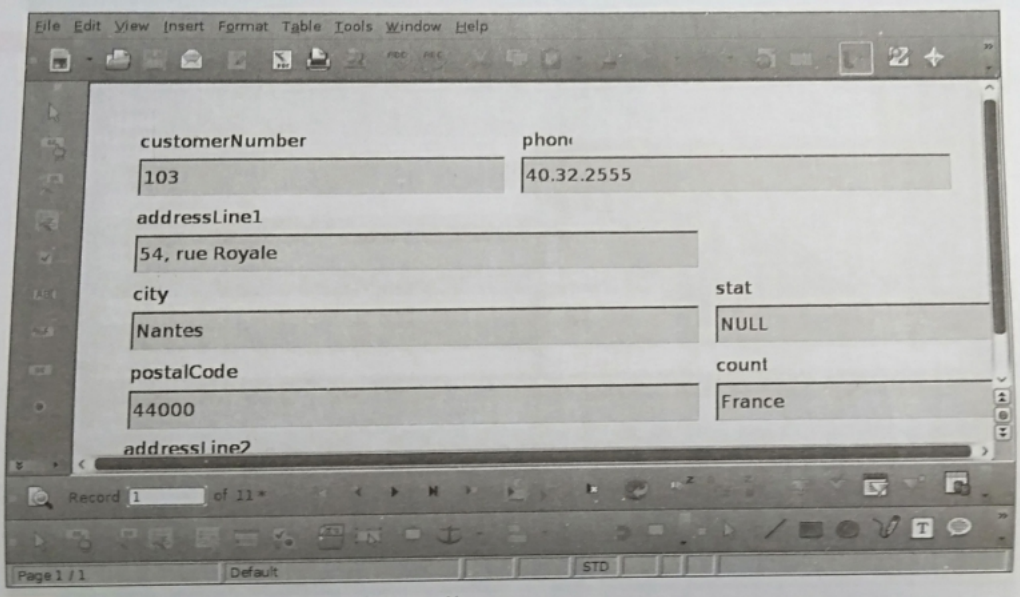


FIGURE 11.8 Form for changing data in some fields of the Customers table.

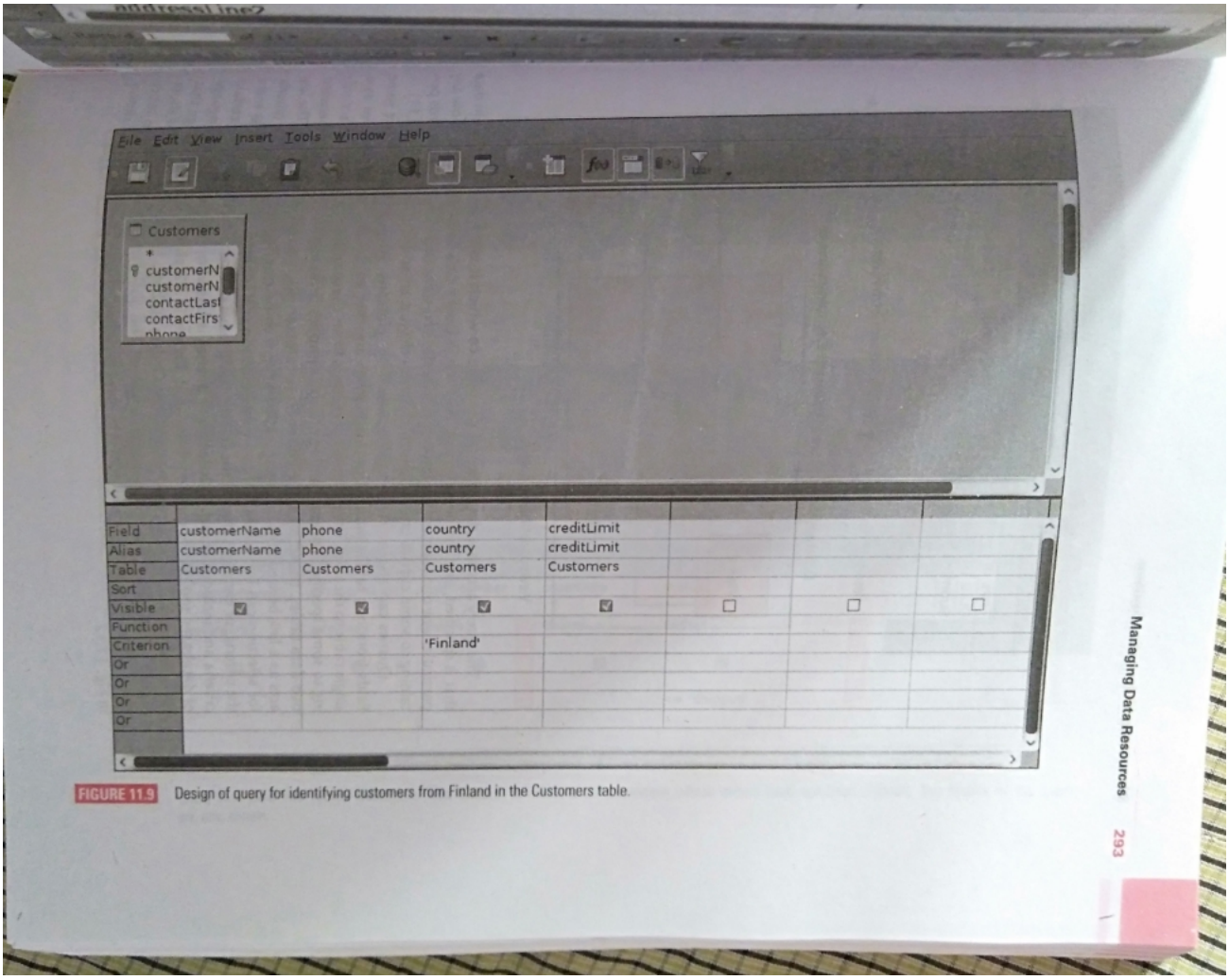


FIGURE 11.9 Design of query for identifying customers from Finland in the Customers table.

File Edit View Insert Tools Window Help

customerName	phone	country	creditLimit
Toys of Finland, Co.	90-224 85	Finland	96500
Oulu Toy Supplies, I	981-44365	Finland	90500
Suominen Souvenie	+358 9 80	Finland	98800

FIGURE 11.10 Results of running the query about customers from Finland on the Customers table.

Different Views Reveal Different Combinations of Data

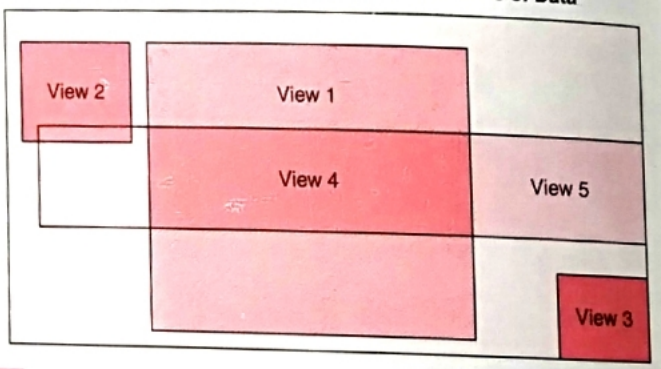


FIGURE 11.11 Views of data using queries.

The query acts as a data filter on the table to which it is applied. It keeps in data that is specified, and removes the rest. As such, a query can be designed to show any particular *view* of data from the database. The desired view can include all records pertaining to some criteria or portions of records relevant to some criteria (Fig.11.11).

Queries can also be run on tables *joined* by relations. A query can extract some fields from one table, some other fields from another table and present them as a single table to the user. The selection of records from these tables can be done according to some specified criteria. Joining of tables implies drawing a relationship between two data tables, based on some common parameter, such as a primary key. The key acts as a link between the tables and allows them to be queried together. The criteria for selection can be inclusive, that is, the selection will include all records that meet the criteria, or exclusive, that is, the selection will leave out records that meet a certain criteria and include the rest. The design of queries is based on the relational language of databases and can be quite sophisticated. Figure 11.12 shows a query design that joins two tables, the 'Customers' and the 'Orders' tables, to obtain a list of those customers whose orders have not been shipped. The query uses '<>' symbol to select those values not equal to 'Shipped'. The data obtained from the query is also shown in Fig. 11.12.

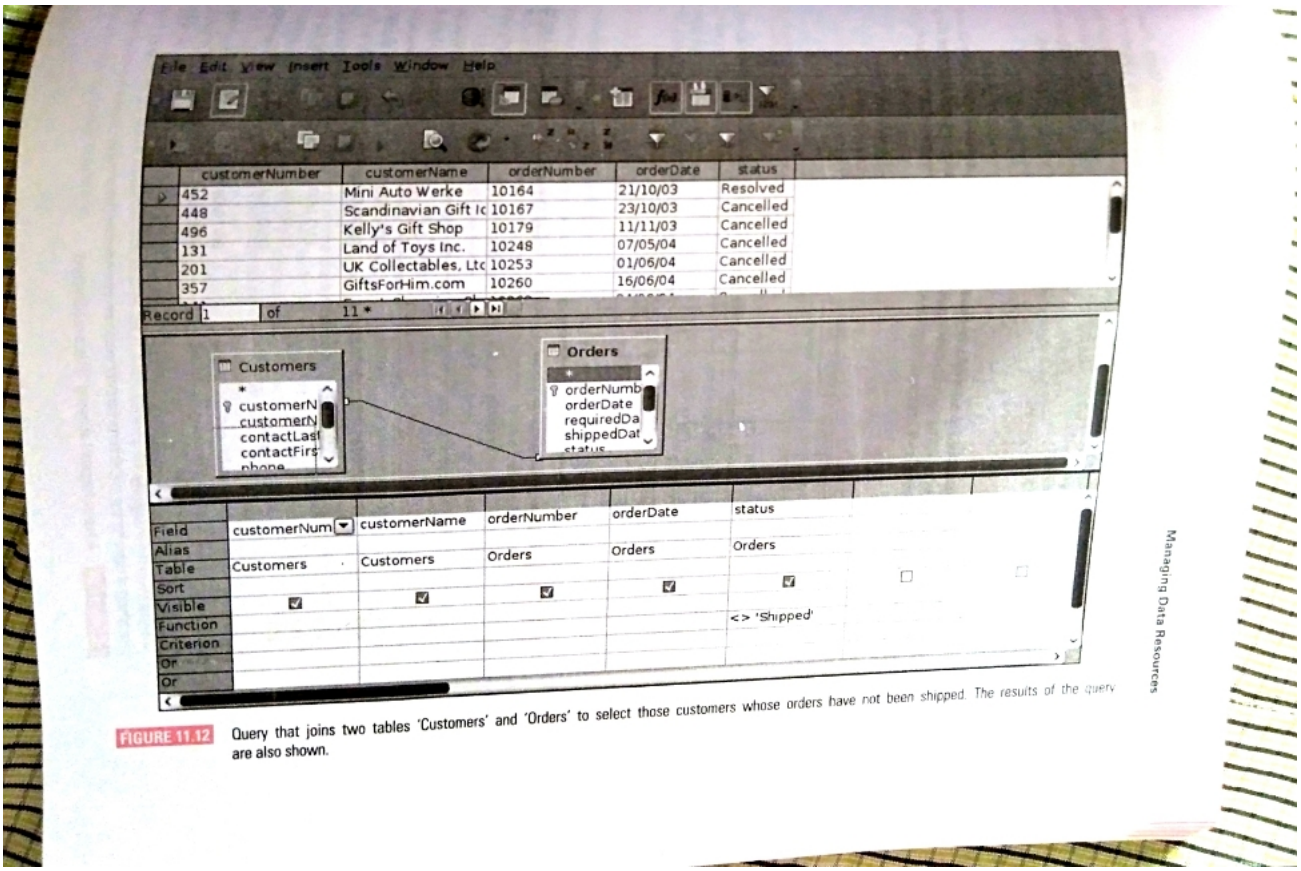


FIGURE 11.12 Query that joins two tables 'Customers' and 'Orders' to select those customers whose orders have not been shipped. The results of the query are also shown.

The above examples show that two tables can be related to extract data. However, in relational databases more than two tables can be related. It is usually the case that all the tables in a database are related in some manner. The relationships depend on the business problem to be solved. The manner in which tables are related in a database is called *schema*. An example schema for the Classic Models database is shown in Fig. 11.13. It consists of all the tables and the relations between them.

11.4.1.4 Reports

Another facility provided by a DBMS is that of creating reports. These reports are similar to queries in that they present selected data from tables. In reports, the data is presented in a useful and readable format (see Fig. 11.14). Reports are also used to

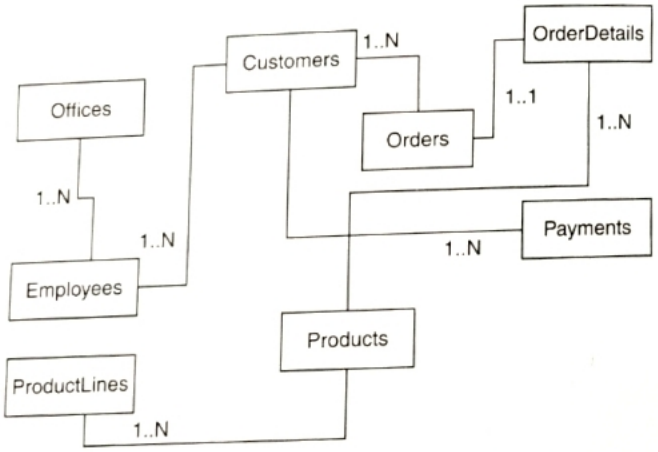


FIGURE 11.13 Schema for Classic Models database. The numbers of the relations indicate cardinality (covered later).

Customer Number	Customer Name	Order Number	Order Date	Status
452	Mini Auto Werke	10164	10/21/03	Resolved
448	Scandinavian Gift Ideas	10167	10/23/03	Cancelled
496	Kelly's gift Shop	10179	11/11/03	Cancelled
131	Land of Toys Inc.	10248	05/07/04	Cancelled
201	UK Collectables, Ltd.	10253	06/01/04	Cancelled
357	GiftsForHim.com	10260	06/16/04	Cancelled
141	Euro + Shopping Channel	10262	06/24/04	Resolved
145	Danish Wholesale Imports	10327	11/10/04	

FIGURE 11.14 A report fragment showing 'Orders not Shipped'

File:	Customers
Prepared by:	R. Hirani
Date:	10-2-2010
Approved by:	V. Chopra
Date:	11-2-2010
Owned by:	Marketing
Master:	Clients
Access:	Sales data entry Op, marketing Op, accounts receivable manager
Data Element:	ID
Description:	Customer identification number
Other names:	None
Value range:	1000–9999
Data type:	Alphanumeric

FIGURE 11.15 Data dictionary for the Customers table.

do calculations and computations on the data in tables. For example, a report can draw on data from incomes and tax schedules, and present the total tax liability in a neat format. Reports also have facilities to include charts to display data graphically. This enables creating highly presentable documents.

11.4.1.5 Metadata

Information regarding the data in tables – for what purpose it is created, by whom, who maintains them and so on – is referred to as *metadata*. This information shows how the tables are structured and what kind of data is stored in them. The latter apsect, which focuses on the data rather than the table structures, is also known as a *data dictionary*. For a typical table, say, the Customers table, a data dictionary is shown in Fig. 11.15.

Metadata is very important for managing and sharing data. Image files, for instance, carry metadata about the image content, the type of image, when it was captured, when it was modified and so on. This data allows the user to adequately store the image so that it can be retrieved for appropriate applications. Metadata also points to how the data is structured and how it can be processed. Metadata is very useful when data has to be migrated from one database to another. For example, if a customerName field data has to be migrated from one database that allows 50 characters to store the data to another database that allows only 40 characters, then the migration program can determine an appropriate way to truncate 50 characters to 40. Metadata helps identify where and how to modify the data to remove 10 extra characters.

11.4.2 SQL

SQL is an acronym for Structured Query Language, a language that is used to define and manipulate data in database tables. SQL was developed along with the relational database concept. Currently, there are various versions of SQL, all of which conform to an international standard.

The most common use of SQL is to query databases. The basic structure of an SQL language query expression consists of three clauses:

1. `SELECT`: The clause lists the desired fields that have to be included in the query.
2. `FROM`: The clause lists the tables from where the data has to be drawn.
3. `WHERE`: The clause specifies the values of the fields that have to be included, or the conditions that have to be met to include the field. Consider the simple query below:

```
SELECT customerName, phone, country
FROM Customers
WHERE country = Finland
```

This query will include the fields '`customerName`' and '`phone`' from the Customers table and select those records whose value for the '`country`' field is Finland. In this example, the `WHERE` clause uses the '=' operator. Other operators that can be used are '>' (greater than) or '<' (less than) or '<>' (not equal to). These operators can be used with numerical values, as shown in the following example:

```
SELECT customerName, phone, creditLimit
FROM Customers
WHERE creditLimit > 50000
```

Queries with SQL can also be used to join two tables to extract data from both the tables, based on some conditions. The join is defined by some common element in both the tables. As an example, consider the following SQL commands:

```
SELECT Customers.customerNumber, Customers.customerName,
Orders.orderNumber, Orders.orderDate, Orders.status
FROM Orders, Customers
WHERE Orders.customerNumber = Customers.customerNumber
```

The commands above use the '`.`' notation to indicate which fields are to be selected from which table. So, the '`Customers.customerNumber`' indicates that the `customerNumber` field is to be selected from the Customers table. The '`status`' field is, as shown, taken from the Orders table. The `WHERE` clause states that all the records collected by the `SELECT` clause must be such that the `customerNumber` from both the tables match. This ensures that only those records are selected that conform to the same customer in each table. Such an operation in relational languages is known as a *join*.

SQL clauses can also relate more than two tables. To extend the example above, it is possible to write a command that relates three or even more tables together, given that all the tables have one field common with one other table (to create the join).

SQL also allows for data definition – creating tables with commands, and data manipulation – adding and updating data in tables. Data definition uses commands such as *Create*, *Alter*, *Truncate* and *Drop*; each command does approximately what its English language meaning suggests. Data manipulation commands include *Insert*, *Update* and *Delete* used to add and modify data in tables.

Although the graphical user interface used by databases, such as Base and Access, precludes the need for SQL-type commands, the SQL commands are a

simple and compact way of doing the same things. Often, SQL commands are used to do some very complicated tasks that would be hard to execute using a graphical interface. The SQL language provides a powerful means to access highly complex databases with perhaps thousands of tables that are related through thousands of relations. Many expert database programmers often use only the SQL commands to manage commercial databases and leave the graphical interface for casual users.

The SQL commands are also the means by which application programs call the database and read, update and modify data. Extensions of SQL, called Procedural Stored Modules, allow programmers creating applications in other languages, such as C or Java, to write SQL commands that will seamlessly integrate the application with the data from the tables. These languages allow sophisticated logic to be used to manipulate tables and data, consistent with the needs of the application.

11.4.3 Commercial DBMS

There are many commercial DBMS available in the market, both as proprietary and as open source products. Each product has captured a market share owing to some unique feature. Each of them also uses a version, or dialect, of SQL. Table 11.2 lists some commercial DBMS along with their features.

11.4.4 E–R Diagrams

The task of a database design is often accomplished by creating Entity–Relationship (E-R) Models. These models provide a high-level view of the data requirements of an organisation. A set of diagrammatic tools are used to create such models, and these are known as entity-relationship diagrams or E–R diagrams.

Table 11.2 Features of Commercial DBMS

Product	Features
Oracle	Released in 1979, this is one of the earliest commercial relational DBMS in the market. It has been the most widely used database, with a 44% market share reported in 2007. Oracle has the distinction of being the first commercial DBMS that popularised the relational database concept of using distributed tables for the first time and also used the SQL standard. Oracle also supports the largest table size (the number of records in a table) among all databases. This product remains the best option for enterprise-wide databases, providing a complete range of DBMS features.
DB2	DB2 is a DBMS popularised by IBM, one of the pioneers of computing in the world. DB2, along with Oracle, is one of the first to implement SQL. DB2 has a strong presence in the enterprise database market although it has versions that can be run as personal and workgroup databases. In 2007, DB2 had a market share of 21%.
SQL Server	This product is offered by Microsoft Corporation and has the third highest market share of 18.5%. This product is largely a workgroup database, meant for smaller departmental or functional applications rather than the enterprise.

(Continued)

Table 11.2

MySQL	MySQL is an open source database, as it was first developed using open source principles, and one version of it still remains in the public domain. MySQL has a commercial version that is now owned by the Oracle corporation, and this had a market share of less than 1%. However, the open source version has a massive market share in terms of installations, as MySQL powers many websites across the world, including famous ones such as Wikipedia, Google and Facebook. Furthermore, MySQL is used extensively by many open source projects that require a database, and it has a presence in thousands of products. MySQL is also used extensively by enterprise customers.
PostgreSQL	This is an object-relational DBMS that is also open source in nature. Its commercial market share is negligible (as it is not sold directly), but it has a very strong presence in many online sites such as gaming sites, Skype, Yahoo! and MySpace. PostgreSQL is considered to be one of the most advanced implementations of a DBMS.
Teradata	Teradata DBMS was started in 1979 and was one of the first to create what is now known as data warehouse. Teradata's speciality is that of very large databases that are run on high-end hardware. Teradata had a market share of about 4% in 2007.

E-R Models are descriptions of the business requirements of data. They are not the actual, physical data model (of tables and relationships) that is maintained in databases, but they are the first step in designing and creating a physical data model. E-R Models are used to extract the business requirements of data and the rules by which data are related. In this sense they are similar to Data Flow Diagrams, with the difference that they focus on the need for and use of data. E-R Models essentially focus on the user side of systems and model how the user sees and will use the system. The users may belong to different groups and departments, such as customers, employees and students. They may have different needs to transact and use data. When their needs are documented within E-R diagrams, it is important to note that this is one of the views of the database that will be visible to them. In this sense, E-R Models help arrive at a possible design, which can then be elaborated upon to create an entire schema.

An E-R diagram essentially consists of data *entities*, *relationships* between entities and *attributes* that describe the entities and relationships. A data entity is any object, person, place, concept or transaction (or event) on which data has to be maintained by the organisation, for which the analysis is being conducted. It is usually easy to identify basic entities for any organisation as they pertain to the very core of the activities organisations are engaged in. For a business organisation, entities such as customers, orders, products, employees, departments, warehouses and stores would be included quite naturally. Similarly, for a university, the most natural entities would be students, professors, courses and classrooms among others.

Entities are described by attributes (see Fig 11.16). Each entity has to be described by at least one attribute. The attributes are properties or facts about the entity. For example, a customer will have attributes such as name, address, phone, identity number and so on. There is no limit on the number of attributes that can be used to describe an entity. The number of attributes used is dependent on the business

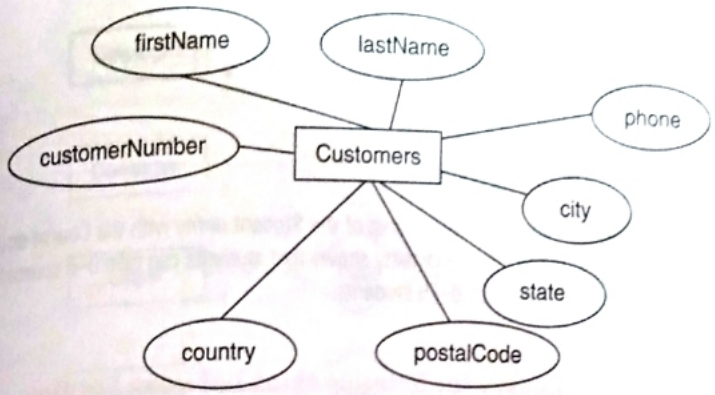


FIGURE 11.16 Customers entity with attributes.

requirements of the organisation. For instance, if an organisation contacts its customers only via digital networks then there is no need for it to maintain attributes related to the physical address of the customer.

Of the attributes used to describe an entity, at least one should be unique valued. This attribute will constitute the primary key of the table described in the previous sections. This unique-valued attribute may be generated by the system, or may be selected according to the requirements of the organisation. For example, universities may choose to use the roll number of students as the primary key of the student entity so as to display this value on reports.

Attributes may require a value (i.e. be forced to have one, such as the name of a customer) or may not (such as the phone number of a customer). The latter are called *optional* attributes. Attributes may also be single-valued or multi-valued. For example, the first name of a customer may be multi-valued (allowing the customer to enter one or more middle names), whereas the system may ensure a single-valued last name for the customer. Furthermore, attributes may be directly stored in the table, or may be derived from other values. For example, the tax liability of an employee may be stored directly in the attribute, or may be computed from the salary value.

Relationships identify natural links or associations between entities. For E–R Modelling, they are used to show how entities have to be treated and how data from different entities can be associated. For example, an entity called 'Student' can be related to an entity called 'Course'. The natural association between the two is that students take courses. When associated in this manner, a question arises: Do students in the entity take all or many or one or none of the courses in the course entity? This question raises the issue of *cardinality*, or the number of entities in a relationship.

A 'Student' entity will consist of many *instances* of students (e.g. Aamir Khan, Shahrukh Khan, etc.). Similarly, the 'Course' entity will consist of many instances of courses (Physics, Chemistry, etc.). The cardinality of a relationship shows how many students, minimum and maximum, can take how many courses. The maximum and minimum number of the cardinality is determined by the business context. For example, a university may specify that a student may take no courses, or may take

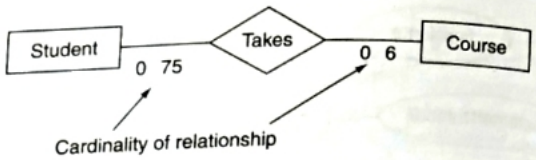


FIGURE 11.17 E–R diagram showing relationship of the Student entity with the Course entity. The relationship is 'takes'. The cardinality shows that students can take 0–6 courses, while each course can be taken by 0–75 students.

Table 11.3 Types of Cardinality for Relationships between Entities

Cardinality	Relationship
0 N to 0 N	Zero to many instances of one table can be related to none or many instances of another table.
1 to 1	One, and only one, instance of one table can be related to one of another table.
0 N to 1	Zero to many instances of one table can be related to exactly one instance of another table.
N to N	Many instances of one table can be related to many instances of another table.

a maximum of six courses. Similarly, the university may specify that each course can be taken by no student (i.e. it has no enrolment) or may be taken by a maximum of 75 students (see Fig. 11.17). Should a student be listed as a student instance of the 'Student' entity even if he/she has not taken any course? This question has to be answered in the context of the university concerned. Many universities may allow a student to remain as a student even though the person is not enrolled in any courses. Similarly, the database may maintain an entity instance of 'Course' even though the course has no students enrolled.

Cardinality for relationships between entities can be of many types, as illustrated in Table 11.3. Some examples of the various types of relationships between entities are depicted in Fig. 11.18.

Some relationships become quite complex as they are explained within the modelling process. For instance, the relationship called 'Enrols' designates the entity 'Student' taking a 'Course'. Enrolment itself is a complex transaction – it is done at a particular date, it may require payment of fees, it may have requirements of pre-requisites and so on. In such situations, where the relationship requires maintaining data, it is often converted to an entity. This requires careful analysis at the database design stage.

11.4.5 Normalisation

While designing databases, it is important to keep in mind the issue of redundant or repeating data. Redundant data creeps into the system if the design of the database is such that it does not explicitly prevent it. Normalisation is the process of table design where groups of fields (or attributes) are kept together. Redundancy is removed by

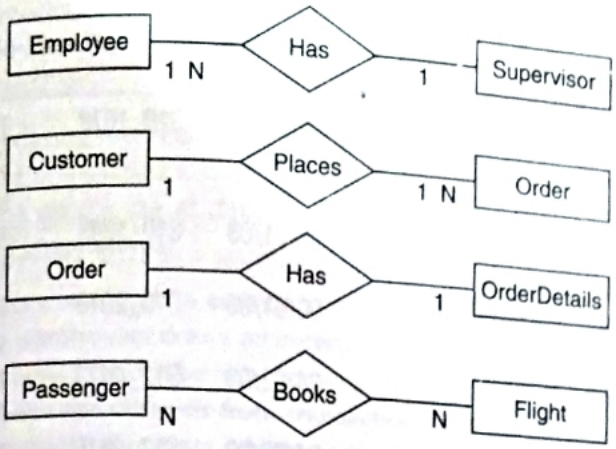


FIGURE 11.18 Different types of relationships between entities.

Table 11.4 Data for an Invoice

customerName	customerNumber	orderNumber	orderDate	productCode	productName
Atelier graphique	103	10100	06/01/03	S10_1678	1969 Harley Davidson Ultimate Chopper
				S10_1949	1952 Alpine Renault 1300
				S10_2016	1996 Moto Guzzi 1100i
Dragon Souveniers, Ltd.	148	10107	24/02/03	S12_4473	1957 Chevy Pickup
				S12_4675	1969 Dodge Charger
				S18_1097	1940 Ford Pickup Truck

this process as only the most pertinent fields are kept in a table and the others that are needed are used from other tables.

For example, consider Table 11.4. It contains data that is used to create an invoice for the Classic Models dataset. The two customers in the table – Atelier graphique and Dragon Souveniers Ltd – have placed one order with multiple products. For each order an order number and date is recorded and then the product codes and names are listed. For each customer and order number there are three products in the invoice. It is not possible to maintain a table in such a form in a database, as each field has to have a single value.

To avoid multiple entries in the product code and product name fields, the table is now converted to the form as shown in Table 11.5. The values for the customer's name and number and also for the order number and date have been inserted in the rows for the two other products in the same order. Now, each row in the table is

Table 11.5 Data Converted to 1NF

customerName	customerNumber	orderNumber	orderDate	productCode	productName
Atelier graphique	103	10100	06/01/03	S10_1678	1969 Harley Davidson Ultimate Chopper
Atelier graphique	103	10100	06/01/03	S10_1949	1952 Alpine Renault 1300
Atelier graphique	103	10100	06/01/03	S10_2016	1996 Moto Guzzi 1100i
Dragon Souveniers, Ltd.	148	10107	24/02/03	S12_4473	1957 Chevy Pickup
Dragon Souveniers, Ltd.	148	10107	24/02/03	S12_4675	1969 Dodge Charger
Dragon Souveniers, Ltd.	148	10107	24/02/03	S18_1097	1940 Ford Pickup Truck

identified by two keys, the customer number and the order number. However, it will be clear that the rows are not uniquely identified by these keys alone.

The first table is converted to the second one, to bring it to a form known as the First Normal Form or 1NF. A table is said to be in 1NF when it contains no repeating entries in any field. The process of converting the table to 1NF is the first step in normalisation.

The table above, though, presents several challenges. If a customer calls and asks for a change in the order, or cancels an order, it is possible that the data is entered incorrectly (say the name of the customer is misspelt or an incorrect product code is entered for the product). To avoid such possible problems, it is best to further break up the table so that only the most relevant changes need be made. One way to do this would be to have three separate tables: one for customers, one for orders and one for products. These are further steps in normalisation of tables to reduce redundancy.

Data normalisation is done to convert tables to second and third normal (2NF and 3NF) and beyond. Currently, the tables can be converted up to the sixth normal form too, however, this depends on the context of the application in which the database is being used.

Though normalisation is useful and reduces redundancy and consequently anomalies and discrepancies in data, it is also leads to inefficient processing. Normalisation is very useful for maintaining and updating data, however, for using data in an organisation it is best to have the most relevant data in a single place. Thus, many database managers also denormalise data (i.e. aggregate separate tables) so that constant use is easier and requires less processing.

11.5 DATA WAREHOUSES

Since the inception of desktop computing, in the mid-1980s, around the world, there has been a proliferation of data use and needs for data storage. Almost all employees of organisations, above a certain size, now use computers and produce, modify or

read data. For very large organisations, the amount of data that is used on a day-to-day basis could be as high as in petabytes. With this huge explosion in data, organisations felt the need for:

1. Consolidating much of the data from various databases into a whole that could be understood clearly.
2. Focusing on the use of data for decision making, as opposed to simply for running transactions.

The need for creating data *warehouses* arose from the above two needs. The technology of data warehouses draws on enterprise databases to create a separate set of tables and relations that can be used to run particular kinds of queries and analytical tools. Warehouses are different from transaction databases, as users can run complex queries on them, which are related to the functions of the enterprise that need not affect the transaction processing.

To create a data warehouse, data is extracted from transactional tables and preprocessed to remove unwanted data types and then loaded into tables in the warehouse. The extraction process requires making queries into transactional databases that are currently being used. This is a challenge as the data tables may be distributed across various servers, and the data may be changing rapidly. The data obtained from these tables is maintained in a *staging* area, a temporary storage area, where the data is *scrubbed*. The idea of data scrubbing is to remove clearly recognisable erroneous data. This task is often difficult, as errors are not obvious – say a misspelt name or a wrong address – and require careful examination to remove them. At the scrubbing stage, data is not corrected in any manner; it is invariably removed from the collection of raw data.

Once the data is scrubbed or cleaned, it is loaded onto the tables that constitute the warehouse. When an organisation is creating a warehouse for the first time, the entire data is loaded into a database, using a particular design. Subsequent data that is obtained from the transaction databases is then extracted, cleaned and loaded incrementally to the earlier tables.

Data pertaining to a particular domain or a problem to be analysed is maintained in data *marts*. For example, a mart may be created to examine sales data alone. This mart will collect data related to the sales activities across the organisation and store them in the warehouse. However, it will exclude the data related to production, finance, employees and so on. The mart can then be analysed for particular problems related to the sales trends, sales predictions and so on. Furthermore, the mart may be updated on a periodic basis to include the fresh data available.

Data in warehouses can be stored in tables with timestamps. This is the *dimensional* method of creating warehouses. The idea here is to store data in a single or a few, unrelated tables that are given one additional attribute of a timestamp (that indicates when the data was collected or created). For example, one table in a dimensional warehouse may include data on customers, sales, products, orders, shipping and a timestamp of each transaction. Each timestamp will pertain to one particular event in the life of the organisation when a transaction occurred and the data was created. Such a table can be analysed to examine trends in sales, fluctuations in orders across seasons and so on.

Another method of storing data is in the regular tables-and-relations format of relational databases. Here too an additional attribute of a timestamp is included within the tables.

Various kinds of analysis can be conducted on data available in warehouses including data mining, online analytical processing and data visualisation. These different methods are designed to extract patterns and useful information from very large data sets. Online analytical process (OLAP) is used to analyse and report data on sales trends, forecasts and other time-series-based analyses. Such analyses allow managers to see and visualise the data in a form that shows interesting and unusual patterns that would not be easily visible from the analysis of transaction data alone.

In modern organisations, ones that have a strong online presence and collect data from customer visits to websites and transaction data from different types of e-commerce sites, the extent and size of the data is such that analysing it for patterns is almost impossible, unless a warehouse is used. For example, one firm analyses data, using OLAP, from millions of visitors to different pages of its website to dynamically place advertisements that would conform to the visitors' interests, as determined by the regions on the page the visitor hovers over or clicks on.

Data warehouses are an active area of development and have strong commercial potential for database vendors. Almost all major commercial vendors of DBMS have products that can be used to create and manage warehouses.

11.5.1 Data Mining Uses

Data mining means extracting patterns and knowledge from historical data that is typically housed in data warehouses. Data mining is a loose term that means many things at the same time – data storing, data analysis, use of artificial intelligence techniques, statistical pattern recognition and others. The original idea of data mining came from the field known as 'knowledge discovery in databases' (KDD). KDD is a sub-field of artificial intelligence and is concerned with finding useful, human-understandable knowledge from data. Several other terms are now used to describe the same ideas – business intelligence, data analytics and web analytics. These concepts are covered in Chapter 13.

Data mining is used with data accumulated in data warehouses. Following are some examples of data stored in warehouses that are used for mining:

1. **Click-stream data:** This data is collected from website pages as users click on links or other items on the web page. Data on where a user clicks, after what interval, what page the users goes to, does the user return and visit other links, etc., are collected. The data are mined to identify which links are most frequently visited, for how long and by what kind of users. The online search firm, Google, has initiated an entire field of mining click-stream data that is known as web analytics.
2. **Point-of-sale purchase data:** Data obtained from retail sales counters is the classic data set to which mining software was applied. The data pertains to the item quantities, price values, date and time of purchase, and details about customers that are obtained from point-of-sale terminals. The data is used to perform 'market basket' analysis, which essentially shows what kinds of items are typically

purchased with each other. In a famous case, a large retailer found from a market basket analysis that men in a certain part of the USA were likely to buy beer and diapers on Thursday evenings. This was an unusual finding and the retailer sought to explain why. It was later learned that many families with young children planned their weekend trips on Thursday evening, at which point the women would ask men to go and buy diapers from the store. The men would take this opportunity to buy beer, also for the weekend. The retailer used this information to announce promotions and increase sales of both these products.

3. **Online search data:** This data is about search that users type in search boxes on web pages. Many organisations collect the text typed in by users while they are searching for some information. This text data reveals what users are interested in and is mined for patterns. The data collected pertains to the search texts typed in, the time at which they are typed and the number of times different searches are done. Many online retailers, such as Flipkart, mine this data to identify what users are interested in and then make product suggestions based on association rules. For example, users searching for books on Java programming may be offered what others have seen and purchased, including associated books on programming they have not considered.
4. **Text data:** This is text data that is posted by users on web pages, blogs, e-mails, wikis, twitter feeds and others. Many organisations have found that by mining this data they can glean interesting insights and trends. Many tools and software programs have been created recently to mine text data. One example is provided by the online site called Wordle.

FIGURE 11.19 Wordle cloud for text from part III of the Constitution of India pertaining to Fundamental Rights.

The www.wordle.net site hosts an application that mines text submitted to it. The application counts the frequency of words appearing in the submitted text and then creates a word 'cloud' with the most frequent words appearing as the largest. For example, Fig. 11.19 has a word cloud of the text of a portion of the Constitution of India. Text from Part III of the Constitution, comprising of the Fundamental Rights, was submitted to wordle.net. This part consists of about 13 pages of printed text.

Chapter Glossary

Database A software program that enables storage, access and use of data by other software applications.

Field A defined space of given size, which stores the basic elements of data.

Record A collection of fields.

File A collection of records, also called a table.

Metadata Contains details about how data is stored in files; provides information on how the data can be used and managed.

Primary key A field that contains data that uniquely identifies a record.

Personal database Databases created and used primarily by single users.

Database server A database software that runs on an independent computer and provides services to client applications.

Distributed database A database whose tables are maintained on various different servers.

Middleware Software that is used to connect distributed databases to different client devices.

Relational database A database whose tables are associated through special links called relations.

Database management systems (DBMS) Software package that has tools for creating, updating, accessing and managing data within a database.

Forms Special objects used within DBMS to update data in tables.

Queries Special objects used within DBMS to access data from tables.

Reports Special objects used within DBMS to present data and analysis in an appealing manner.

Joins Relations created between tables.

Data dictionary A special data structure to store metadata.

SQL A high-level computer language to create, update, access and manage databases.

Entity Any object, place, person, concept or transaction on which data is maintained in a database.

Attributes Properties of entities.

Cardinality The order of a relationship between two entities.

Data warehouse A massive database that is created from organisational data to enable analysis and to assist with decision making.

Data scrubbing Removing erroneous data from data warehouses.

Data mart A data warehouse containing data pertaining to a particular domain.

Review Questions

1. What is the need for data management? Why is it difficult to manage data?
2. Describe some of the challenges of modern database management.
3. What is the difference between fields, records and files?
4. Why is a primary key needed?
5. What is the difference between a personal database and an organisational database?
6. What is the advantage of three-tier architecture?
7. Why is middleware important?
8. Is it possible to query two or more tables at the same time? How?
9. Why is it important to maintain metadata?
10. What are three basic SQL commands and what do they achieve?

Research Questions

1. Study the library database in your university or institute. Try to draw a schema of the basic entities of the database and the relations between them.
2. Seek out examples of how organisations use data stored in warehouses. What decision making issues are the organisations trying to address?
3. Search the Internet for the concept of 'Big Data'. Find out what it means and how organisations deal with this kind of data.
4. Download and install Open Office or Libre Office on a personal computer you have access to. Then dowload the Birt database that is discussed in the text (the source is also mentioned). Recreate all the forms and queries in the text. Then create a few queries on your own.

Further Reading

1. An article 'How much Information is there in the World?' in USC News, February 2011 Is available at:
http://uscnews.usc.edu (accessed on June 2011).
2. An article 'eBay's Two Enormous Data Warehouses', in DBMS2, 2009 is available at:
http://www.dbms2.com/2009/04/30/ebays-two-enormous-data-warehouses/ (accessed on June 2011).
3. Hoffer, J.A., Prescott, M.B. and McFadden, F.R. (2007) *Modern Database Management*, 8th edn, Prentice Hall, NJ.