Chapter 1. Introduction to Database Systems

UNIT I

1.1. Introduction

In today's competitive environment, data (or information) and its efficient management is the most critical business objective of an organisation. It is also a fact that we are in the age of information explosion where people are bombarded with data and it is a difficult task to get the right information at the right time to take the right decision. Therefore, the success of an organisation is now, more than ever, dependent on its ability to acquire accurate, reliable and timely data about its business or operation for effective decision-making process.

Database system is a tool that simplifies the above tasks of managing the data and extracting useful information in a timely fashion. It analyses and guides the activities or business purposes of an organisation. It is the central repository of the data in the organisation's information system and is essential for supporting the organisation's functions, maintaining the data for these functions and helping users interpret the data in decision-making. Managers are seeking to use knowledge derived from databases for competitive advantages, for example, to determine customer buying pattern, tracking sales, support customer relationship management (CRM), on-line shopping, employee relationship management, implement decision support system (DSS), managing inventories and so on. To meet the changing organisational needs, database structures must be flexible to accept new data and accommodate new relationships to support the new decisions.

With the rapid growth in computing technology and its application in all spheres of modern society, databases have become an integral component of our everyday life. We encounter several activities in our day-to-day life that involve interaction with a database, for example, bank database to withdraw and deposit money, air or railway reservation databases for booking of tickets, library database for searching of a particular book, supermarket goods databases to keep the inventory, to check for sufficient credit balance while purchasing goods using credit cards and so on.

In fact, databases and database management systems (DBMS) have become essential for managing our business, governments, banks, universities and every other kind of human endeavour. Thus, they are a critical element of today’s software industry to support these requirements and a daunting task to solve the problems of managing huge amounts of data that are increasingly being stored.

This chapter introduces the basic concepts of databases and database management system (DBMS), reviews the goals of DBMS, types of data models and storage management system.

1.2. Basic Concepts and Definitions

With the growing use of computers, the organisations are fast migrating from a manual system to a computerised information system for which the data within the organisation is a basic resource. Therefore, proper organisation and management of data is necessary to run the organisation efficiently. The efficient use of data for planning, production control, marketing, invoicing, payroll, accounting and other function in an organisation have a major impact for its competitive edge. In this section, formal definition of the terms used in databases is provided.

1.2.1. Data

Data may be defined as a known fact that can be recorded and that have implicit meaning. Data are raw or isolated facts from which the required information is produced.

Data are distinct pieces of information, usually formatted in a special way. They are binary computer representations of stored logical entities. A single piece of data represents a single fact about something in which we are interested. For an industrial organisation, it may be the fact that Thomas Mathew’s employee (or social security) number is 106519, or that the largest supplier of the casting materials of the organisation is located in Indore, or that the telephone number of one of the key customers M/s Elbee Inc. is 001-732-3931650. Similarly, for a Research and Development set-up it may be the fact that the largest number of new products as on date is 100, or for a training institute it may be the fact that largest enrolment were in Database Management course. Therefore, a piece of data is a single fact about something that we care about in our surroundings.

Data can exist in a variety of forms that have meaning in the user’s environment such as numbers or text on a piece of paper, bits or bytes stored in computer’s memory, or as facts stored in a person’s mind. Data can also be objects such as documents, photographic images and even video segments. The example of data is shown in Table 1.1.

Table 1.1. Example of data

In Salesperson’s view	In Electricity supplier’s context	In Employer’s mind
Customer-name	Consumer-name	Employee-name
Customer-account	Consumer-number	Identification-number
Address	Address	Department
Telephone numbers	Telephone numbers	Date-of-birth
	Unit consumed	Qualification

Table 1.1. Example of data

In Salesperson’s view	In Electricity supplier’s context	In Employer’s mind
	Amount-payable	Skill-type

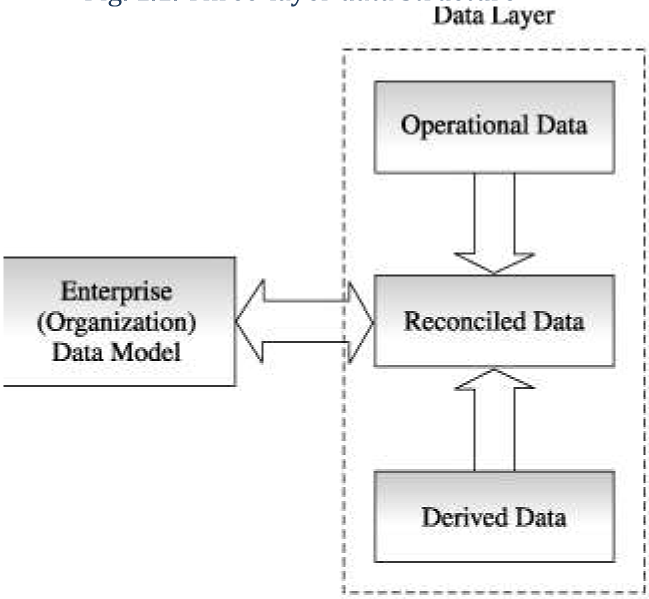
Usually there are many facts to describe something of interest to us. For example, let us consider the facts that as a Manager of M/s Elbee Inc., we might be interested in our employee Thomas Mathew. We want to remember that his employee number is 106519, his basic salary rate is Rs. 2,00,000 (US$ 4000) per month, his home town is Jamshedpur, his home country is India, his date of birth is September 6th, 1957, his marriage anniversary is on May 29th, his telephone number is 0091-657-2431322 and so forth. We need to know these things in order to process Mathew’s payroll check every month, to send him company greeting cards on his birthday or marriage anniversary, print his salary slip, to notify his family in case of any emergency and so forth. It certainly seems reasonable to collect all the facts (or data) about Mathew that we need for the stated purposes and to keep (store) all of them together. Table 1.2 shows all these facts about Thomas Mathew that concern payroll and related applications.

Table 1.2. Thomas Mathew’s payroll facts

Employee Number	Employee Last Name	Employee First Name	Basic Salary (US$)	Home Town	Home Country	Date of Birth	Marriage Anniversary	Telephone Number
106519	Mathew	Thomas	4000	Jamshedpur	India	06-09-1957	29.05	0091-657-2431322

Data is also known as the plural of datum, which means a single piece of information. However, in practice, data is used as both-the singular and the plural form of the word. The term data is often used to distinguish machine-readable (binary) information from human-readable (textual) information. For example, some applications make a distinction between data files (that contain binary data) and text files (that contain ASCII data). Either numbers, or characters or both can represent data.

Fig. 1.1. Three-layer data structure



Operational data are stored in various operational systems throughout the organisation (both internal and external) systems.

Reconciled data are stored in the organisation data warehouse and in operational data store. They are detailed and current data, which is intended as the single, authoritative source for all decision support applications.

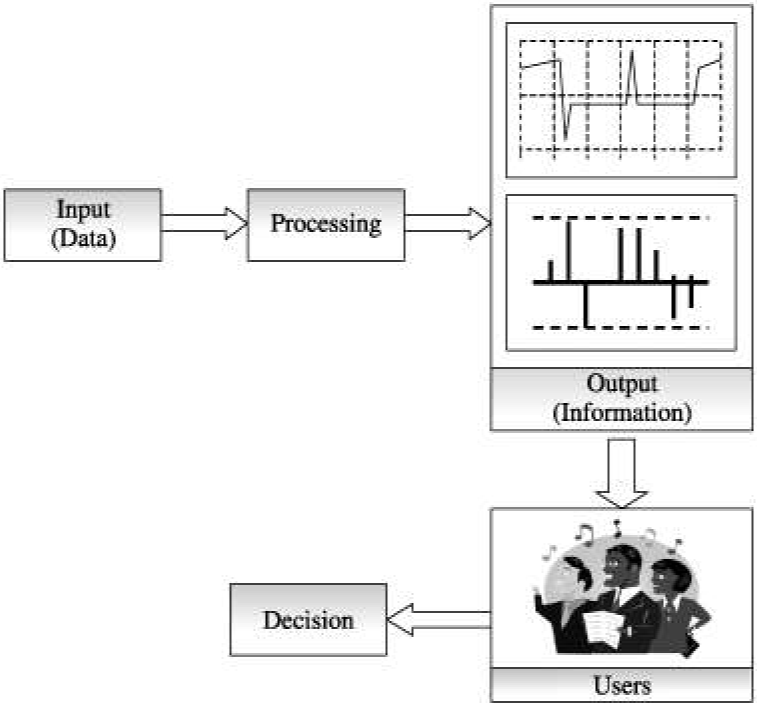
Derived data are stored in each of the data mart (a selected, limited and summarised data warehouse). Derived data are selected, formatted and aggregated for end-user decision support applications.

1.2.2. Information

Data and information are closely related and are often used interchangeably. Information is processed, organised or summarised data. It may be defined as collection of related data that when put together, communicate meaningful and useful message to a recipient who uses it, to make decision or to interpret the data to get the meaning.

Data are processed to create information, which is meaningful to the recipient, as shown in Fig. 1.2. For example, from the salesperson's view, we might want to know the current balance of a customer M/s Waterhouse Ltd., or perhaps we might ask for the average current balance of all the customers in Asia. The answers to such questions are information. Thus, information involves the communication and reception of knowledge or intelligence. Information apprises and notifies, surprises and stimulates. It reduces uncertainty, reveals additional alternatives or helps in eliminating irrelevant or poor ones, influences individuals and stimulates them into action. It gives warning signals before some thing starts going wrong. It predicts the future with reasonable level of accuracy and helps the organisation to make the best decisions.

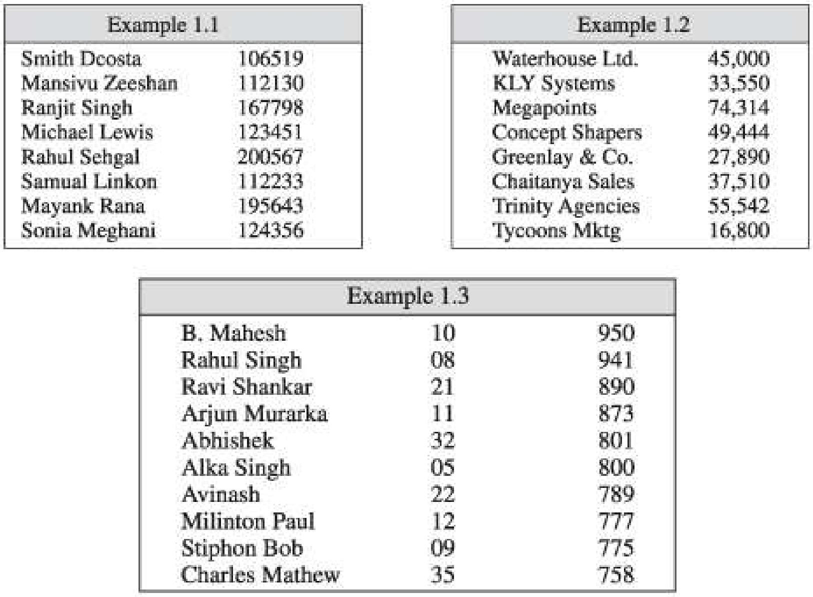
Fig. 1.2. Information cycle



1.2.3. Data Versus Information

Let us take the following two examples with the given list of facts or data as shown in Fig. 1.3. Both the examples given below 1.1, 1.2 and 1.3 satisfy the definition of data, but the data are useless in their present form as they are unable to convey any meaningful message. Even if we guess in example 1.1 that it is person's names together with some identification or social security numbers, that in example 1.2 it is customer's names together with some money transaction and in example 1.3 it may be student's name together with the marks obtained in some examination the data remain useless since they do not convey any meaning about the purpose of the entries.

Fig. 1.3. Data versus Information



Example 1.1	
Smith Dcosta	106519
Mansivu Zeeshan	112130
Ranjit Singh	167798
Michael Lewis	123451
Rahul Sehgal	200567
Samual Linkon	112233
Mayank Rana	195643
Sonia Meghani	124356

Example 1.2	
Waterhouse Ltd.	45,000
KLY Systems	33,550
Megapoints	74,314
Concept Shapers	49,444
Greenlay & Co.	27,890
Chaitanya Sales	37,510
Trinity Agencies	55,542
Tycoons Mktg	16,800

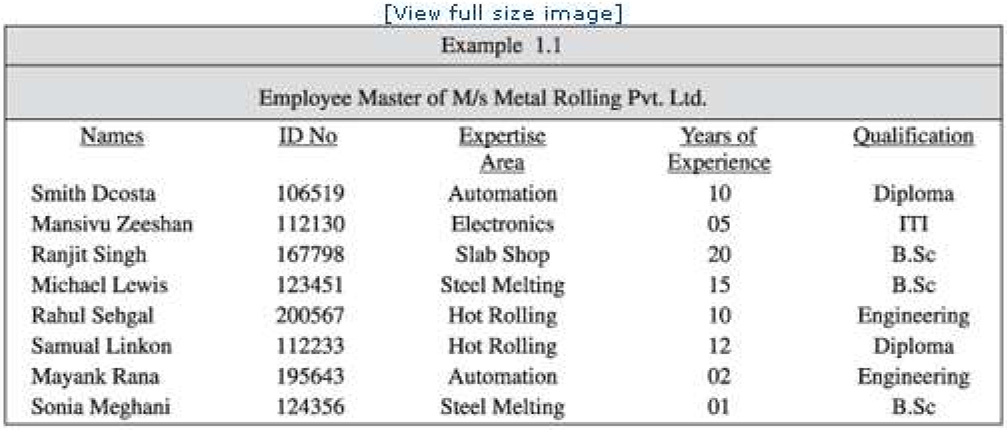
Example 1.3		
B. Mahesh	10	950
Rahul Singh	08	941
Ravi Shankar	21	890
Arjun Murarka	11	873
Abhishek	32	801
Alka Singh	05	800
Avinash	22	789
Milinton Paul	12	777
Stiphon Bob	09	775
Charles Mathew	35	758

Now let us modify the data in example 1.1 by adding a few additional data and providing some structure and place the same data in a context shown in Fig. 1.4 (a). Now data has been rearranged or processed to provide meaningful message or information, which is an Employee Master of M/s Metal Rolling Pvt. Ltd. Now this is useful information for the departmental head or the organisational head for taking decisions related to the additional requirement of experienced and qualified manpower.

Fig. 1.4. Converting data into information for Example 1.1

(a) Data converted into textual information

[View full size image]



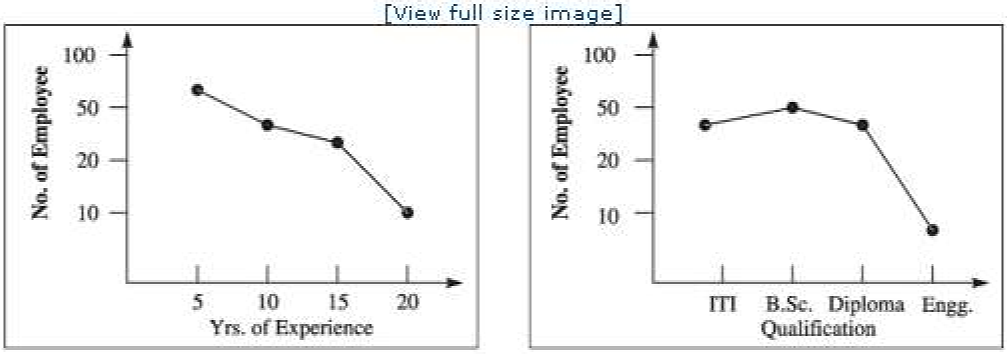
Example 1.1

Employee Master of M/s Metal Rolling Pvt. Ltd.

Names	ID No	Expertise Area	Years of Experience	Qualification
Smith Dcosta	106519	Automation	10	Diploma
Mansivu Zeeshan	112130	Electronics	05	ITI
Ranjit Singh	167798	Slab Shop	20	B.Sc
Michael Lewis	123451	Steel Melting	15	B.Sc
Rahul Sehgal	200567	Hot Rolling	10	Engineering
Samual Linkon	112233	Hot Rolling	12	Diploma
Mayank Rana	195643	Automation	02	Engineering
Sonia Meghani	124356	Steel Melting	01	B.Sc

(b) Data converted into summarised information

[View full size image]

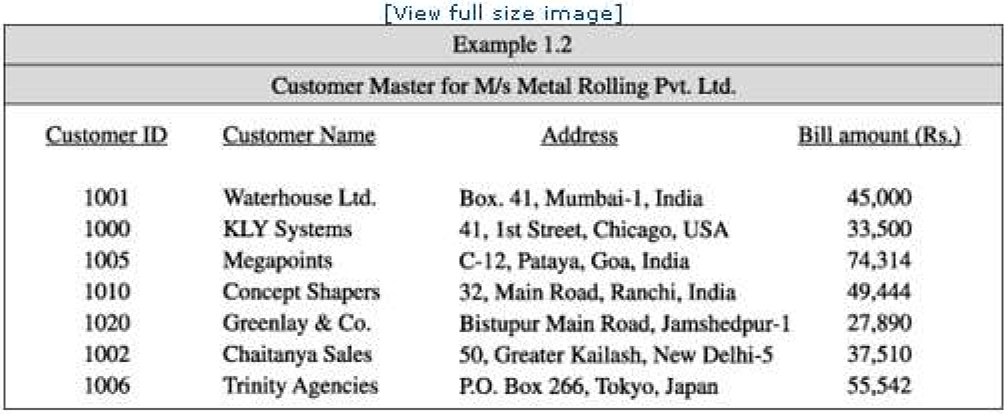


Another way to convert data intoformation is to summarise them or otherwise process and present them for human interpretation. For example, Fig. 1.4 (b) shows summarised data related to the number of employees versus experience and qualification presented as graphical information. This information could be used by the organisation as a basis for deciding whether to add or hire new experienced or qualified manpower.

Data in Example 1.2 can be modified by adding additional data and providing some structure, as shown in Fig. 1.5. Now data has been rearranged or processed to provide meaningful message or information, which is a Customer Invoicing of M/s Metal Rolling Pvt. Ltd. Now this is useful information for the organisation to sending reminders to the customer for the payment of pending balance amount and so on. Similarly, as shown in Fig. 1.6, the data has been converted into textual and summarised information for Example 1.3.

Fig. 1.5. Converting data into information for Example 1.2

[View full size image]



Example 1.2

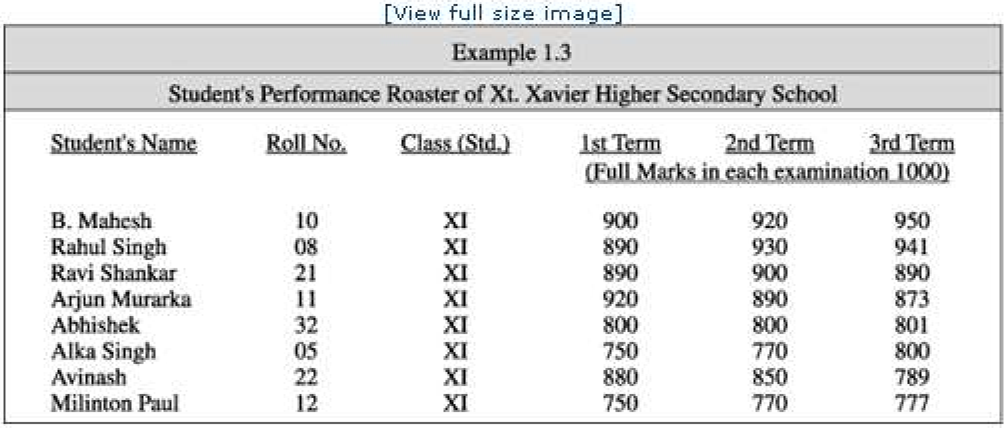
Customer Master for M/s Metal Rolling Pvt. Ltd.

Customer ID	Customer Name	Address	Bill amount (Rs.)
1001	Waterhouse Ltd.	Box. 41, Mumbai-1, India	45,000
1000	KLY Systems	41, 1st Street, Chicago, USA	33,500
1005	Megapoints	C-12, Pataya, Goa, India	74,314
1010	Concept Shapers	32, Main Road, Ranchi, India	49,444
1020	Greenlay & Co.	Bistupur Main Road, Jamshedpur-1	27,890
1002	Chaitanya Sales	50, Greater Kailash, New Delhi-5	37,510
1006	Trinity Agencies	P.O. Box 266, Tokyo, Japan	55,542

Fig. 1.6. Converting data into information for Example 1.3

(*a*) Data converted into textual information

[View full size image]



Example 1.3

Student's Performance Roaster of Xt. Xavier Higher Secondary School

Student's Name	Roll No.	Class (Std.)	1st Term	2nd Term	3rd Term
			(Full Marks in each examination 1000)		
B. Mahesh	10	XI	900	920	950
Rahul Singh	08	XI	890	930	941
Ravi Shankar	21	XI	890	900	890
Arjun Murarka	11	XI	920	890	873
Abhishek	32	XI	800	800	801
Alka Singh	05	XI	750	770	800
Avinash	22	XI	880	850	789
Milinton Paul	12	XI	750	770	777

(*b*) Data converted into summarised information

Today, database may contain either data or information (or both), according to the organisations definition and needs. For example, a database may contain an image of the Employee Master shown in Fig. 1.4 (a) or Customer Master shown in Fig. 1.5 or Student's Performance Roaster shown in Fig. 1.6 (a), and also in summarised (trend or picture) form shown in Figs. 1.4 (b) and 1.6 (b) for decision support functions by the organisation. In this book, the terms data and information have been treated as synonymous.

1.2.4. Data Warehouse

Data warehouse is a collection of data designed to support management in the decision-making process. It is a subject-oriented, integrated, time-variant, non-updatable collection of data used in support of management decision-making processes and business intelligence. It contains a wide variety of data that present a coherent picture of business conditions at a single point of time. It is a unique kind of database, which focuses on business intelligence, external data and time-variant data (and not just current data).

Data warehousing is the process, where organisations extract meaning and inform decision making from their informational assets through the use of data warehouses. It is a recent initiative in information technology and has evolved very rapidly. A further detail on data warehousing is given in Chapter 20.

1.2.5. Metadata

A metadata (also called the data dictionary) is the data about the data. It is also called the system catalog, which is the self-describing nature of the database that provides program-data independence. The system catalog integrates the metadata. The metadata is the data that describes objects in the database and makes easier for those objects to be accessed or manipulated. It describes the database structure, constraints, applications, authorisation, sizes of data types and so on. These are often used as an integral tool for information resource management.

Metadata is found in documentation describing source systems. It is used to analyze the source files selected to populate the largest data warehouse. It is also produced at every point along the way as data goes through the data integration process. Therefore, it is an important by-product of the data integration process. The efficient management of a production or enterprise warehouse relies heavily on the collection and storage of metadata. Metadata is used for understanding the content of the source, all the conversion steps it passes through and how it is finally described in the target system or data warehouse.

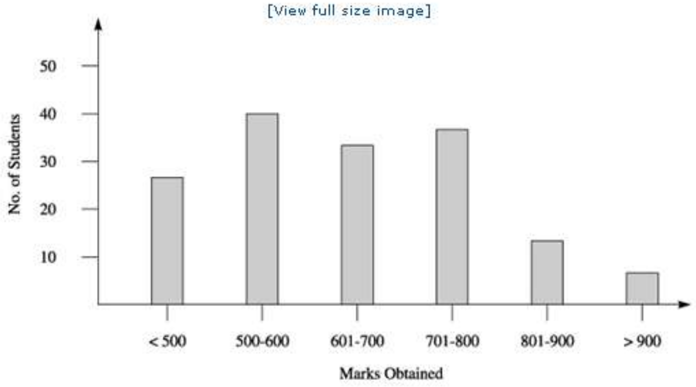
Metadata is used by developers who rely on it to help them develop the programs, queries, controls and procedures to manage and manipulate the warehouse data. Metadata is also used for creating reports and graphs in front-end data access tools, as well as for the management of enterprise-wide data and report changes for the end-user. Change management relies on metadata to administer all of the related objects for example, data model, conversion programs, load jobs, data definition language (DDL), and so on, in the warehouse that are impacted by a change request. Metadata is available to database administrators (DBAs), designers and authorised users as on-line system documentation. This improves the control of database administrators (DBAs) over the information system and the users' understanding and use of the system.

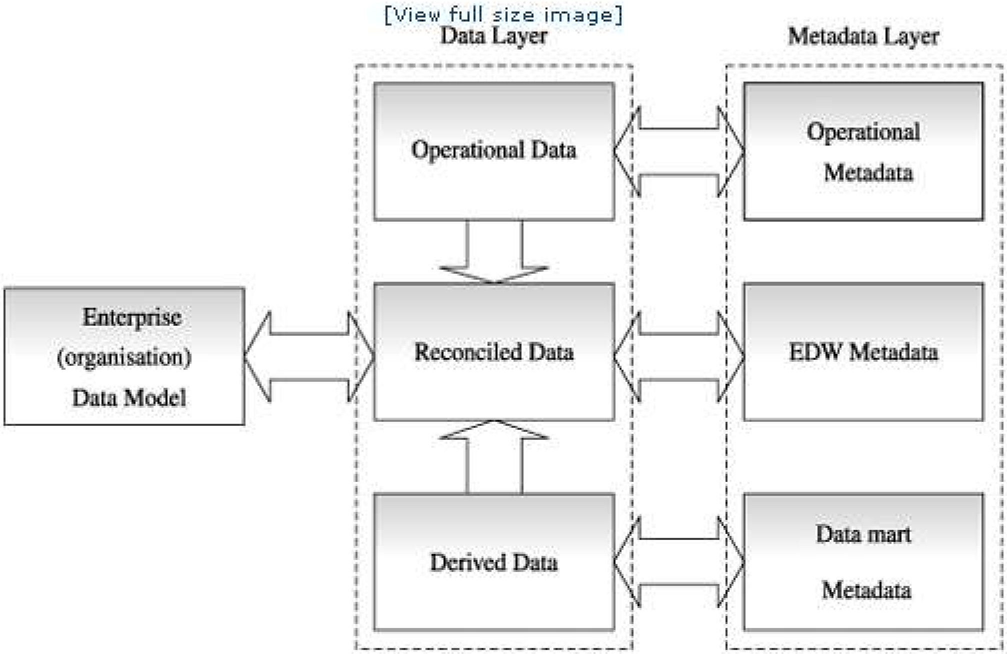
1.2.5.1. Types of Metadata

The advent of data warehousing technology has highlighted the importance of metadata. There are three types of metadata as shown in Fig. 1.7. These metadata are linked to the three-layer data structure as shown in Fig. 1.7.

Fig. 1.7. Metadata layer

[View full size image]





Operational metadata: It describes the data in the various operational systems that feed the enterprise data warehouse. Operational metadata typically exist in a number of different formats and unfortunately are often of poor quality.

Enterprise data warehouse (EDW) metadata: These types of metadata are derived from the enterprise data model. EDW metadata describe the reconciled data layer as well as the rules for transforming operational data to reconciled data.

Data mart metadata: They describe the derived data layer and the rules for transforming reconciled data to derived data.

1.2.6. System Catalog

A system catalog is a repository of information describing the data in the database, that is the metadata (or data about the data). System catalog is a system-created database that describes all database objects, data dictionary information and user access information. It also describes table-related data such as table names, table creators or owners, column names, data types, data size, foreign keys and primary keys, indexed files, authorized users, user access privileges and so forth.

The system catalog is created by the database management system and the information is stored in system files, which may be queried in the same manner as any other data table, if the user has sufficient access privileges. A fundamental characteristic of the database approach is that the database system contains not only the database but also a complete definition or description of the database structure and constraints. This definition is stored in the system catalog, which contains information such as the structure of each file, the type and storage format of each data item and various constraints on the data. The information stored in the catalog is called metadata. It describes the structure of the primary database.

1.2.7. Data Item or Fields

A data item is the smallest unit of the data that has meaning to its user. It is traditionally called a field or data element. It is an occurrence of the smallest unit of named data. It is represented in the database by a value. Names, telephone numbers, bill amount, address and so on in a telephone bill and name, basic allowances, deductions, gross pay, net pay and so on in employee salary slip, are a few examples of data. Data items are the molecules of the database. There are atoms and sub-atomic particles composing each molecule (bits and bytes), but they do not convey any meaning on their own right and so are of little concern to the users. A data item may be used to construct other, more complex structures.

1.2.8. Records

A record is a collection of logically related fields or data items, with each field possessing a fixed number of bytes and having a fixed data type. A record consists of values for each field. It is an occurrence of a named collection of zero, one, or more than one data items or aggregates. The data items are grouped together to form records. The grouping of data items can be achieved through different ways to form different records for different purposes. These records are retrieved or updated using programs.

1.2.9. Files

A file is a collection of related sequence of records. In many cases, all records in a file are of the same record type (each record having an identical format). If every record in the file has exactly the same size (in bytes), the file is said to be made up of fixed-length records. If different records in the file have different sizes, the file is said to be made of variable-length records.

Table 1.3 illustrates an example of a payroll file in tabular form. Each kind of fact in each column, for example, employees number or home-town is called a field. The collection of facts about a particular employee in one line or row (for example, all the fields of all the columns) of the table is an example of record. The collection of payroll facts for all of the employees (all columns and rows), that is, the entire table in Table 1.3 is an example of file.

Table 1.3. Employee payroll file for M/s Metal Rolling Pvt. Ltd.

Employee's Number	Employee's Last Name	Employee's First Name	Basic Salary (US$)	Home Town	Home Country	Date of Birth	Marriage Day	Telephone Number
106519	Mathew	Thomas	4000	Jamshedpur	India	06-09-1957	29.05	2431322
112233	Smith	John	4500	Rome	Italy	16-11-1980	10.12	2423206
123456	Kumar	Rajeev	6000	Delhi	India	20-02-1959	06.06	2427982
123243	Martin	Jose	3500	Mumbai	India	11-12-1965	11.11	2437981
109876	Singh	Abhishek	4800	New York	USA	05-07-1973	07.02	2147008
111222	Parasar	Alka	5100	Detroit	USA	30-09-1979	06.12	2145063

Table 1.3. Employee payroll file for M/s Metal Rolling Pvt. Ltd.

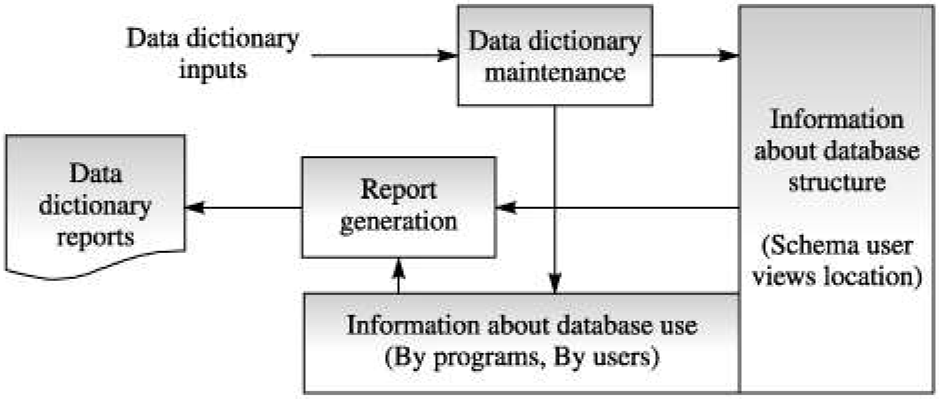
Employee's Number	Employee's Last Name	Employee's First Name	Basic Salary (US$)	Home Town	Home Country	Date of Birth	Marriage Day	Telephone Number
165243	Kumar	Avinash	6500	London	UK	27-05-1966	30.01	2407841

1.3. Data Dictionary

Data dictionary (also called information repositories) are mini database management systems that manages metadata. It is a repository of information about a database that documents data elements of a database. The data dictionary is an integral part of the database management systems (DBMSs) and stores metadata, or information about the database, attribute names and definitions for each table in the database. Data dictionaries aid the database administrator in the management of a database, user view definitions as well as their use.

The most general structure of a data dictionary is shown in Fig. 1.8. It contains descriptions of the database structure and database use. The data in the data dictionary are maintained by several programs and produce diverse reports on demand. Most data dictionary systems are stand-alone systems, and their database is maintained independently of the DBMS, thereby enabling inconsistencies between the database and the data dictionary. To prevent them, the data dictionary is integrated with DBMSs in which the schema and user view definitions are controlled through the data dictionary and are made available to the DBMS software.

Fig. 1.8. Structure of data dictionary


Data dictionary inputs
Data dictionary maintenance
Information about database structure
(Schema user views location)
Data dictionary reports
Report generation
Information about database use
(By programs, By users)

Data dictionary is usually a part of the system catalog that is generated for each database. A useful data dictionary system usually stores and manages the following types of information:

- Descriptions of the schema of the database.
- Detailed information on physical database design, such as storage structures, access paths and file and record sizes.
- Description of the database users, their responsibilities and their access rights.
- High-level descriptions of the database transactions and applications and of the relationships of users to transactions.
- The relationship between database transactions and the data items referenced by them. This is useful in determining which transactions are affected when certain data definitions are changed.

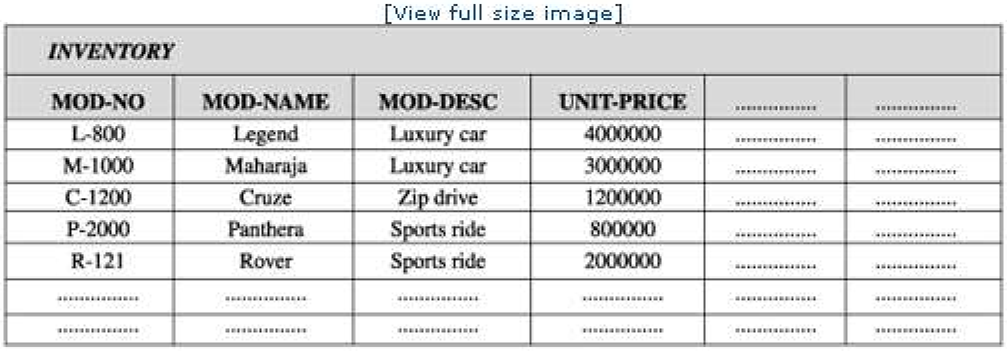
- Usage statistics such as frequencies of queries and transactions and access counts to different portions of the database.

Let us take an example of a manufacturing company M/s ABC Motors Ltd., which has decided to computerise its activities related to various departments. The manufacturing department is concerned with types (or brands) of motor in its manufacturing inventory, while the personnel department is concerned with keeping track of the employees of the company. The manufacturing department wants to store the details (also called entity set) such as the model no., model description and so on. Similarly, personnel department wants to keep the facts such as employee's number, last name, first name and so on. Fig. 1.9 illustrates the two data processing (DP) files, namely INVENTORY file of manufacturing department and EMPLOYEE file of personnel department.

Fig. 1.9. Data processing files of M/s ABC Motors Ltd

(*a*) INVENTORY file of manufacturing department

[View full size image]

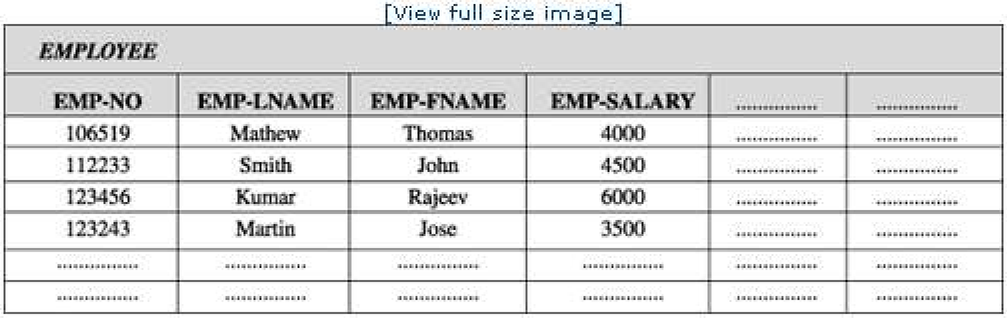


INVENTORY

MOD-NO	MOD-NAME	MOD-DESC	UNIT-PRICE	....	....
L-800	Legend	Luxury car	4000000	....	....
M-1000	Maharaja	Luxury car	3000000	....	....
C-1200	Cruze	Zip drive	1200000	....	....
P-2000	Panthera	Sports ride	800000	....	....
R-121	Rover	Sports ride	2000000	....	....
....	....	....	....	....	....
....	....	....	....	....	....

(*b*) EMPLOYEE file of personnel department

[View full size image]



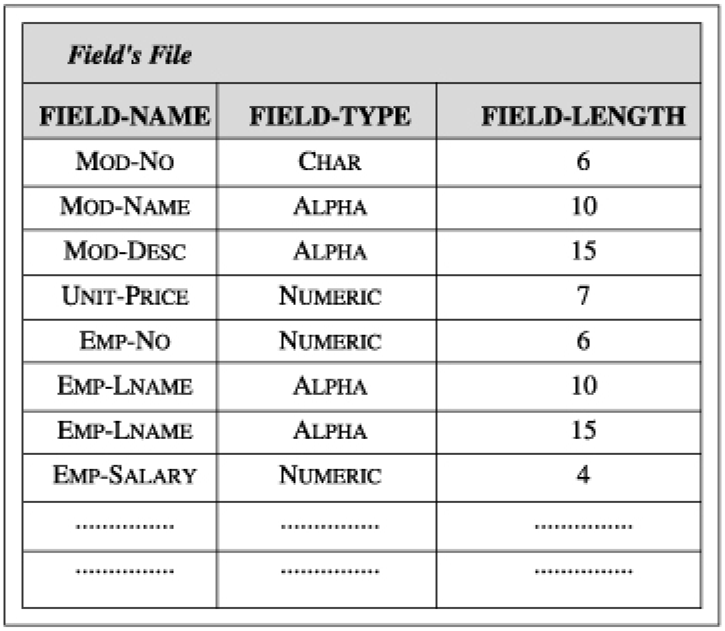
EMPLOYEE

EMP-NO	EMP-LNAME	EMP-FNAME	EMP-SALARY	....	....
106519	Mathew	Thomas	4000	....	....
112233	Smith	John	4500	....	....
123456	Kumar	Rajeev	6000	....	....
123243	Martin	Jose	3500	....	....
....	....	....	....	....	....
....	....	....	....	....	....

Now, though, manufacturing and employee departments are interested in keeping track of their inventory and employees details respectively, the data processing (DP) department of M/s ABC Motors Ltd., would be interested in tracking and managing entities (individual fields and the two files), that is the data dictionary. Fig. 1.10 shows a sample of the data dictionary for the two files (field's file and file's file) of Fig. 1.9.

Fig. 1.10. Data dictionary files of M/s ABC Motors Limited

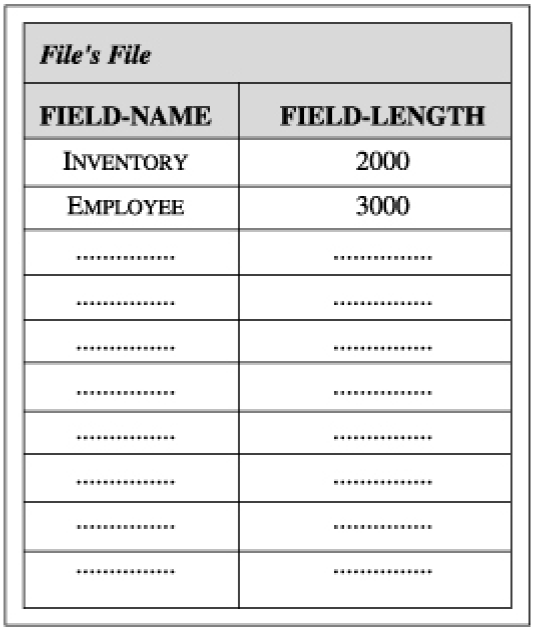
(*a*) Field's file



Field's File

FIELD-NAME	FIELD-TYPE	FIELD-LENGTH
MOD-NO	CHAR	6
MOD-NAME	ALPHA	10
MOD-DESC	ALPHA	15
UNIT-PRICE	NUMERIC	7
EMP-NO	NUMERIC	6
EMP-LNAME	ALPHA	10
EMP-LNAME	ALPHA	15
EMP-SALARY	NUMERIC	4
....	....	....
....	....	....

(*b*) File's file



File's File

FIELD-NAME	FIELD-LENGTH
INVENTORY	2000
EMPLOYEE	3000
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....

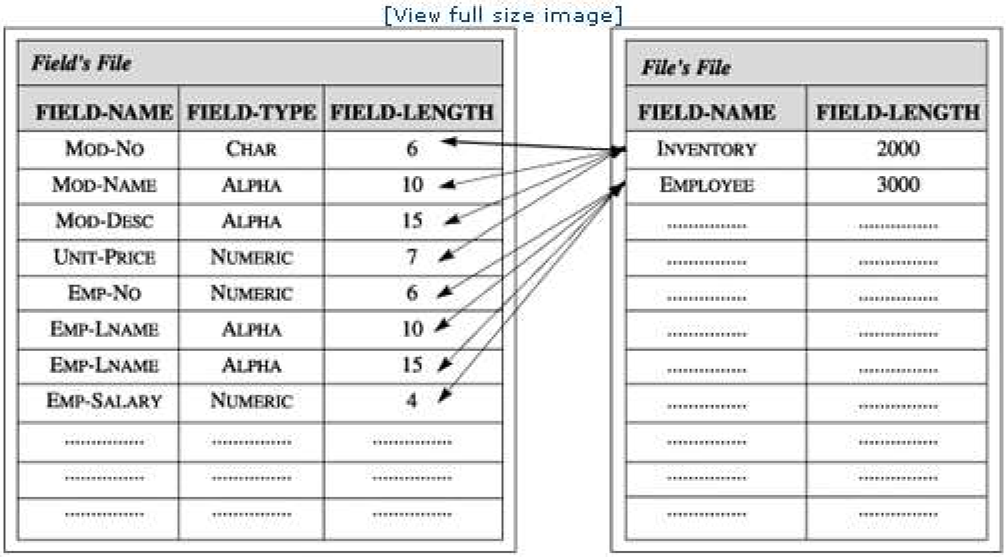
As it can be seen from Fig. 1.10 that all data fields of both the files are included in field's file and both the files (INVENTORY and EMPLOYEE) in the file's file. Thus, the data dictionary contains the attributes for the field's file such as FIELD-NAME, FIELD-TYPE, FIELD-LENGTH and for file's file such as FILE-NAME and FILE-LENGTH.

In the manufacturing department's INVENTORY file, each row (consisting of fields namely MOD-NO, MOD-NAME, MOD-DESC, UNIT-PRICE) represents the details of a model of a car, as shown in Fig. 1.9 (a). In the personnel department's EMPLOYEE file, each row (consisting of fields namely EMP-NO, EMP-LNAME, EMP-FNAME, EMP-SALARY) represents details about an employee, as shown in Fig. 1.9 (b). Similarly, in the data dictionary, each row of the field's file (consisting of entries namely FIELD-NAME, FIELD-TYPE, FIELD-LENGTH) represents one of the fields in one of the application data files (in this case INVENTORY and EMPLOYEE files) processed by the data processing department, as shown in Fig. 1.10 (a). Also, each row of the file's file (consisting of entries namely FILE-NAME, FILE-LENGTH) represents one of the application files (in this case INVENTORY and EMPLOYEE files) processed by data processing department, as shown in Fig. 1.9 (b). Therefore, we see that, each row of the field's file in Fig. 1.10 (a) represents one of the fields of one of the files in Fig. 1.9, and each row of the file's file in Fig. 1.10 (b) represents one of the files in Fig. 1.9.

Data dictionary also keeps track of the relationships among the entities, which is important in the data processing environment as how these entities interrelate. Figure 1.11 shows the links (relationship) between fields and files. These relationships are important for the data processing department.

Fig. 1.11. Data dictionary showing relationships

[View full size image]



Field's File

FIELD-NAME	FIELD-TYPE	FIELD-LENGTH
MOD-NO	CHAR	6
MOD-NAME	ALPHA	10
MOD-DESC	ALPHA	15
UNIT-PRICE	NUMERIC	7
EMP-NO	NUMERIC	6
EMP-LNAME	ALPHA	10
EMP-LNAME	ALPHA	15
EMP-SALARY	NUMERIC	4
....	....	....
....	....	....
....	....	....

File's File

FIELD-NAME	FIELD-LENGTH
INVENTORY	2000
EMPLOYEE	3000
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....
....	....

1.3.1. Components of Data Dictionaries

As discussed in the previous section, data dictionary contains the following components:

- Entities
- Attributes
- Relationships
- Key

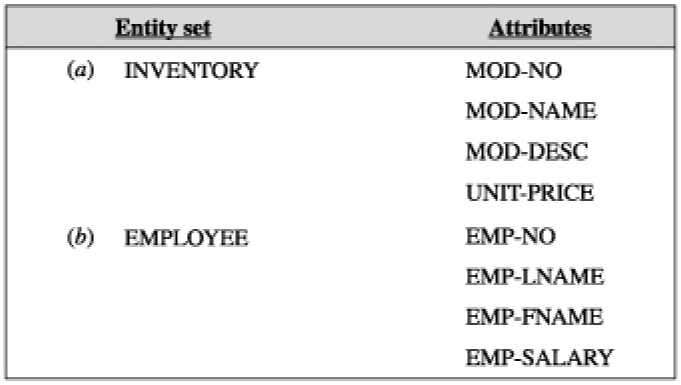
1.3.1.1. Entities

Entity is the real physical object or an event; the user is interested in keeping track of. In other words, any item about which information is stored is called entity. For example, in Fig. 1.9 (b), Thomas Mathew is a real living person and an employee of M/s ABC Motors Ltd., is an entity for which the company is interested in keeping track of the various details or facts. Similarly, in Fig. 1.9 (a), Maharaja model car (Model no. M-1000) is a real physical object manufactured by M/s ABC Motors Ltd., is an entity. A collection of the entities of the same type, for example "all" of the company's employees (the rows in EMPLOYEE file in Fig. 1.9 (b)), and "all" the company's model (the rows in INVENTORY file in Fig. 1.9 (a)) are called an entity set. In other words, we can say that, a record describes the entity and a file describes an entity set.

1.3.1.2. Attributes

An attribute is a property or characteristic (field) of an entity. In Fig. 1.9 (b), Mathew's EMP-NO, EMP-SALARY and so forth, all are his attributes. Similarly, in Fig. 1.9 (a), Maharaja car's MOD-NO, MOD-DESC, UNIT-PRICE and so forth, all are its attributes. In other words, we can say that, values in all the fields are attributes. Fig. 1.12 shows an example of an entity set and its attributes.

Fig. 1.12. Entity set and attributes



Entity set	Attributes
(*a*) INVENTORY	MOD-NO
	MOD-NAME
	MOD-DESC
	UNIT-PRICE
(*b*) EMPLOYEE	EMP-NO
	EMP-LNAME
	EMP-FNAME
	EMP-SALARY

1.3.1.3. Relationships

The associations or the ways that different entities relate to each other is called relationships, as shown in Fig. 1.11. The relationship between any pair of entities of a data dictionary can have value to some part or department of the organisation. Some data dictionaries define limited set of relationships among their entities, while others allow the relationship between every pair of entities. Some examples of common data dictionary relationships are given below:

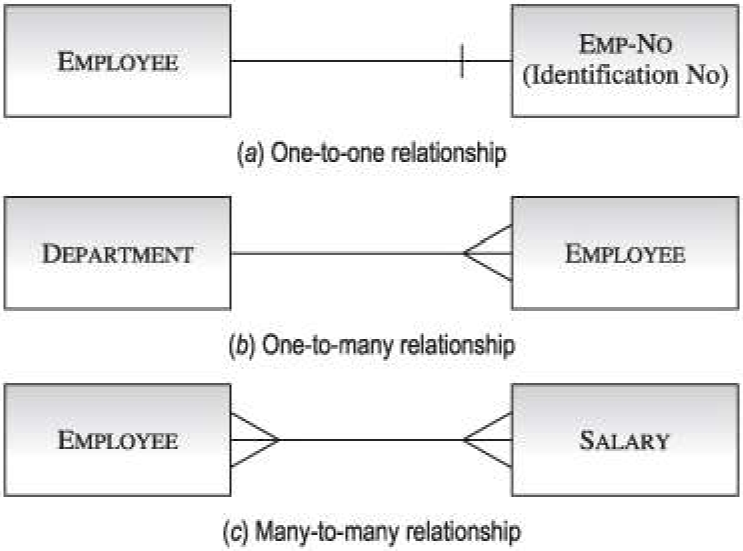
- Record construction: for example, which field appears in which records.
- Security: for example, which user has access to which file.
- Impact of change: for example, which programs might be affected by changes to which files.
- Physical residence: for example, which files are residing in which storage device or disk packs.
- Program data requirement: for example, which programs use which file.
- Responsibility: for example, which users are responsible for updating which files.

Relationships could be of following types:

- One-to-one (1:1) relationship
- One-to-many (1:m) relationships
- Many-to-many (n:m) relationships

Let us take the example shown in Fig. 1.9 (b), wherein there is only one EMP-NO (employee identification number) in the EMPLOYEE file of personnel department for each employee, which is unique. This is called unary associations or one-to-one (1:1) relationship, as shown in Fig. 1.13 (a).

Fig. 1.13. Entity relationship (ER) diagram



(a) One-to-one relationship

(b) One-to-many relationship

(c) Many-to-many relationship

Now let us assume that an employee belongs to a manufacturing department. While for a given employee there is one manufacturing department, in the manufacturing department there may be many employees. Thus, in this case, there is one-to-one relationship in one direction and a multiple association in the other direction. This combination is called one-to-many (1:m) relationship, as shown in Fig. 1.13 (b).

Finally, consider the situation in which an employee gets a particular salary. While for a given employee there is one salary amount (for example, 4000), the same amount may be given to many employees in the department. In this case, there is multiple associations in both the direction, and this combination is called many-to-many (n:m) relationship, as shown in Fig. 1.13 (c).

1.3.1.4. Key

The data item (or field) for which a computer uses to identify a record in a database system is referred to as key. In other words, key is a single attribute or combination of attributes of an entity set that is used to identify one or more instances of the set. There are various types of keys.

- Primary key
- Concatenated key
- Secondary key
- Super key

Primary key is used to uniquely identify a record. It is also called entity identifier, for example, EMP-NO in the EMPLOYEE file of Fig. 1.9 (b) and MOD-NO in the INVENTORY file of Fig. 1.9 (a). When more than one data item is used to identify a record, it is called concatenated key, for example, both EMP-NO and EMP-FNAME in EMPLOYEE file of Fig. 1.9 (b) and both MOD-NO and MOD-TYPE in INVENTORY file of Fig. 1.9 (a).

Secondary key is used to identify all those records, which have a certain property. It is an attribute or combination of attributes that may not be a concatenated key but that classifies the entity set on a particular characteristic. In Super key includes any number of attributes that possess a uniqueness property. For example, if we add additional attributes to a primary key, the resulting combination would still uniquely identify an instance of the entity set. Such keys are called super keys. Thus, a primary key is a minimum super key.

1.3.2. Active and Passive Data Dictionaries

Data dictionary may be either active or passive. An active data dictionary (also called integrated data dictionary) is managed automatically by the database management software. Since active data dictionaries are maintained by the system itself, they are always consistent with the current structure and definition of the database. Most of the relational database management systems contain active data dictionaries that can be derived from their system catalog.

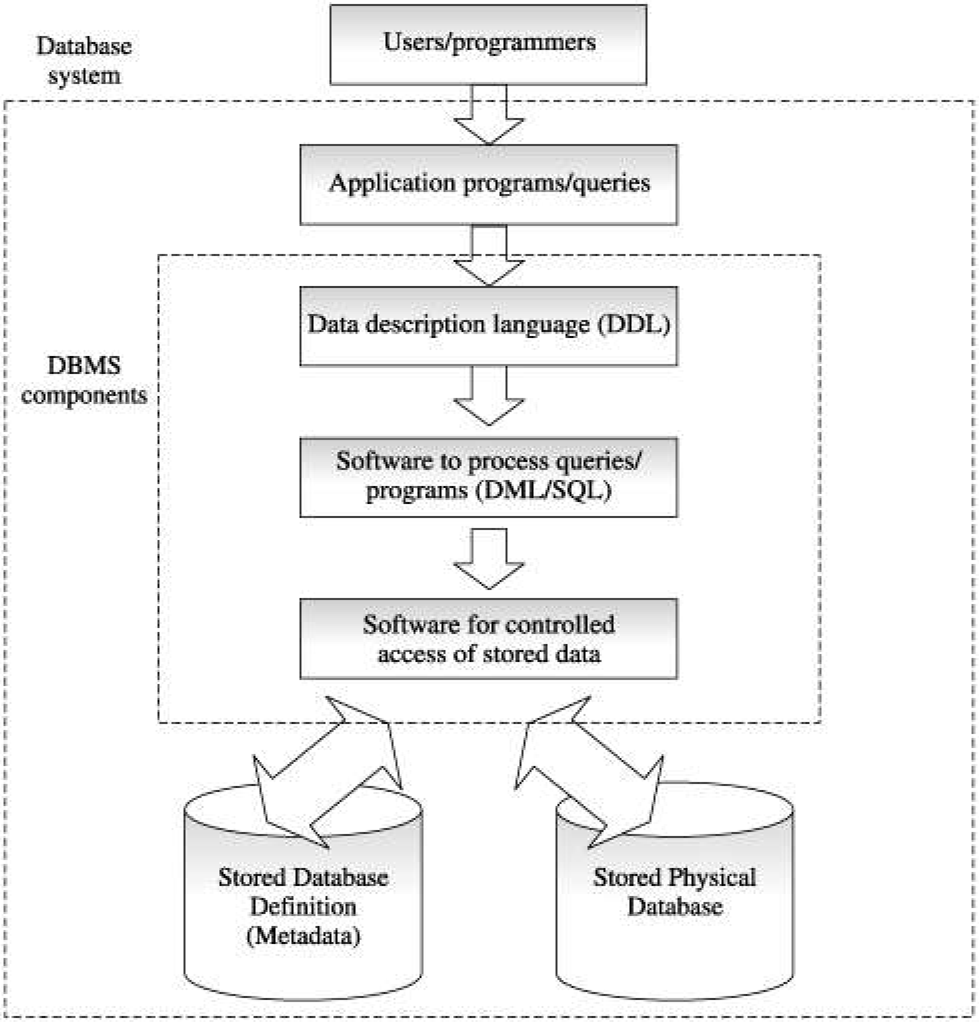
The passive data dictionary (also called non-integrated data dictionary) is the one used only for documentation purposes. Data about fields, files, people and so on, in the data processing environment are entered into the dictionary and cross-referenced. Passive dictionary is simply a self-contained application and a set of files is used for documenting the data processing environment. It is managed by the users of the system and is modified whenever the structure of the database is changed. Since this modification must be performed manually by the user, it is possible that the data dictionary will not be current with the current structure of the database. However, the passive data dictionaries may be maintained as a separate database. Thus, it allows developers to remain independent from using a particular relational database management system for as long as possible. Also, passive data dictionaries are not limited to information that can be discerned by the database management system. Since passive data dictionaries are maintained by the user, they may be extended to contain information about organisational data that is not computerized.

1.5. Database System

A database system, also called database management system (DBMS), is a generalized software system for manipulating databases. It is basically a computerized record-keeping system; which it stores information and allows users to add, delete, change, retrieve and update that information on demand. It provides for simultaneous use of a database by multiple users and tool for accessing and manipulating the data in the database. DBMS is also a collection of programs that enables users to create and maintain database. It is a general-purpose software system that facilitates the process of defining (specifying the data types, structures and constraints), constructing (process of storing data on storage media) and manipulating (querying to retrieve specific data, updating to reflect changes and generating reports from the data) for various applications.

Typically, a DBMS has three basic components, as shown in Fig. 1.16, and provides the following facilities:

Fig. 1.16. DBMS Components



Database system
Users/programmers
Application programs/queries
DBMS components
Data description language (DDL)
Software to process queries/ programs (DML/SQL)
Software for controlled access of stored data
Stored Database Definition (Metadata)
Stored Physical Database

- Data description language (DDL): It allows users to define the database, specify the data types, and data structures, and the constraints on the data to be stored in the database, usually through data definition language. DDL translates the schema written in a source language into the object schema, thereby creating a logical and physical layout of the database.
- Data manipulation language (DML) and query facility: It allows users to insert, update, delete and retrieve data from the database, usually through data manipulation language (DML). It provides general query facility through structured query language (SQL).
- Software for controlled access of database: It provides controlled access to the database, for example, preventing unauthorized user trying to access the database, providing a concurrency control system to allow shared access of the database, activating a recovery control system to restore the database to a previous consistent state following a hardware or software failure and so on.

The database and DBMS software together is called a database system. A database system overcomes the limitations of traditional file-oriented system such as, large amount of data redundancy, poor data control, inadequate data manipulation capabilities and excessive programming effort by supporting an integrated and centralized data structure.

1.5.1. Operations Performed on Database Systems

As discussed in the previous section, database system can be regarded as a repository or container for a collection of computerized data files in the form of electronic filing cabinet. The users can perform a variety of operations on database systems. Some of the important operations performed on such files are as follows:

- Inserting new data into existing data files
- Adding new files to the database
- Retrieving data from existing files
- Changing data in existing files
- Deleting data from existing files
- Removing existing files from the database.

Let us take an example of M/s Metal Rolling Pvt. Ltd. having a very small database containing just one, called EMPLOYEE, as shown in Table 1.4. The EMPLOYEE file in turn contains data concerning the details of employee working in the company. Fig. 1.17 depicts the various operations that can be performed on EMPLOYEE file and the results thereafter displayed on the computer screen.

Table 1.4. EMPLOYEE file of M/s Metal Rolling Pvt. Ltd.

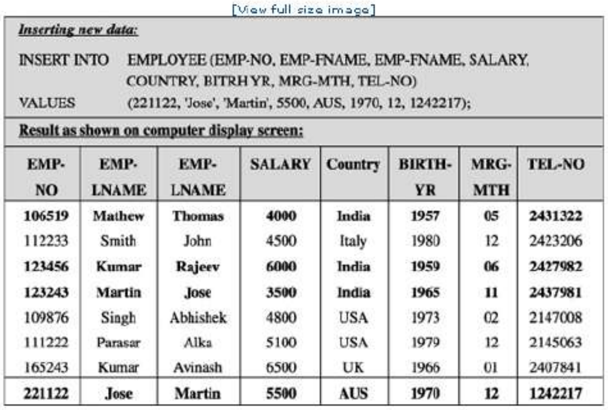
EMPLOYEE

EMP-NO	EMP-LNAME	EMP-FNAME	SALARY	COUNTRY	BIRTH-YR	MRG-MTH	TEL-NO
106519	Mathew	Thomas	4000	India	1957	05	2431322
112233	Smith	John	4500	Italy	1980	12	2423206
123456	Kumar	Rajeev	6000	India	1959	06	2427982
123243	Martin	Jose	3500	India	1965	11	2437981
109876	Singh	Abhishek	4800	USA	1973	02	2147008
111222	Parasar	Alka	5100	USA	1979	12	2145063
165243	Kumar	Avinash	6500	UK	1966	01	2407841

Fig. 1.17. Operations on EMPLOYEE file

(*a*) Inserting new data into a file

[View full size image]



Inserting new data:

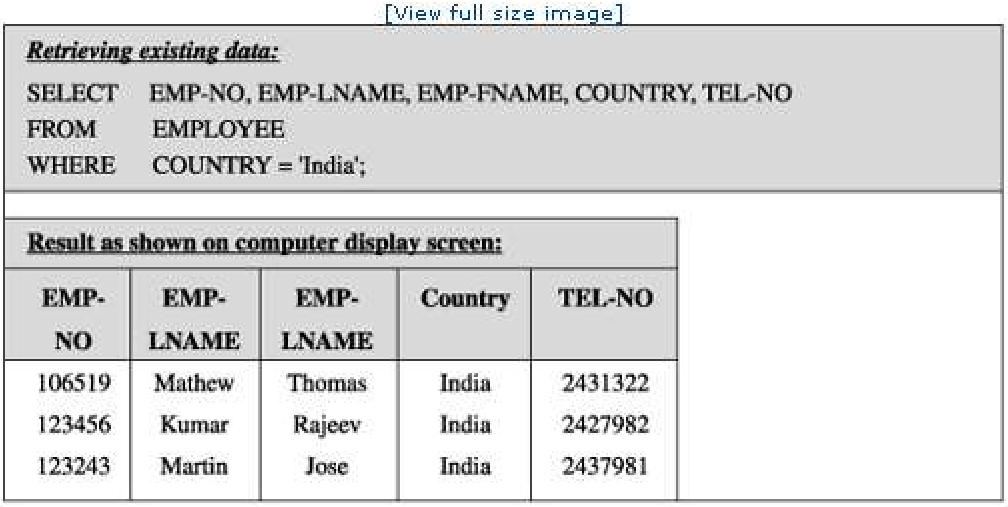
```
INSERT INTO    EMPLOYEE (EMP-NO, EMP-FNAME, EMP-FNAME, SALARY,
               COUNTRY, BITRH YR, MRG-MTH, TEL-NO)
VALUES         (221122, 'Jose', 'Martin', 5500, AUS, 1970, 12, 1242217);
```

Result as shown on computer display screen:

EMP-NO	EMP-LNAME	EMP-LNAME	SALARY	Country	BIRTH-YR	MRG-MTH	TEL-NO
106519	**Mathew**	**Thomas**	**4000**	**India**	**1957**	**05**	**2431322**
112233	Smith	John	4500	Italy	1980	12	2423206
123456	**Kumar**	**Rajeev**	**6000**	**India**	**1959**	**06**	**2427982**
123243	**Martin**	**Jose**	**3500**	**India**	**1965**	**11**	**2437981**
109876	Singh	Abhishek	4800	USA	1973	02	2147008
111222	Parasar	Alka	5100	USA	1979	12	2145063
165243	Kumar	Avinash	6500	UK	1966	01	2407841
221122	**Jose**	**Martin**	**5500**	**AUS**	**1970**	**12**	**1242217**

(*b*) Retrieving existing data from a file

[View full size image]



Retrieving existing data:

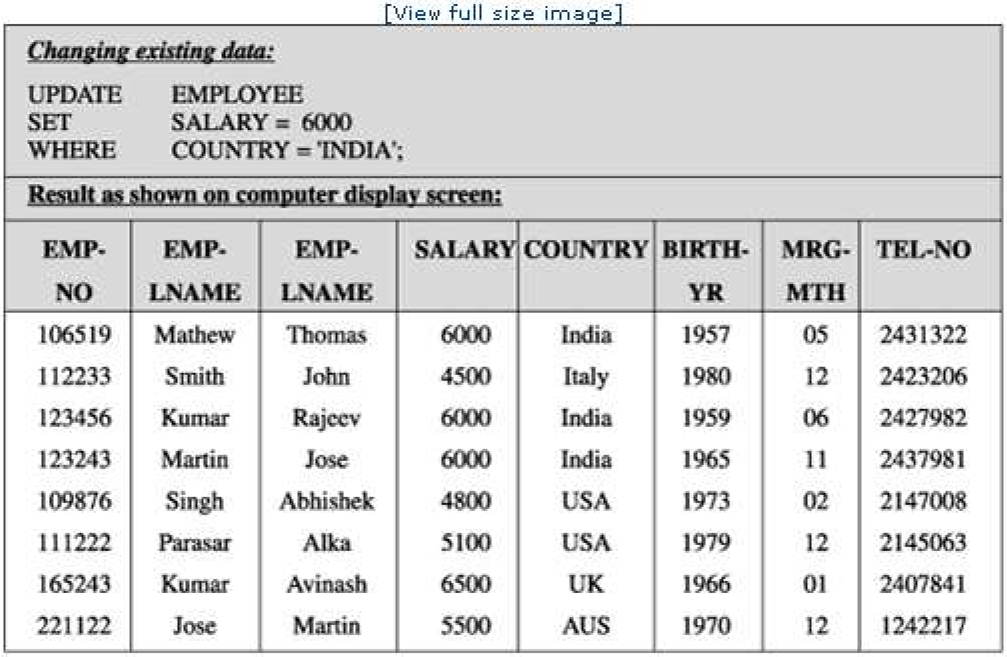
```
SELECT    EMP-NO, EMP-LNAME, EMP-FNAME, COUNTRY, TEL-NO
FROM      EMPLOYEE
WHERE     COUNTRY = 'India';
```

Result as shown on computer display screen:

EMP-NO	EMP-LNAME	EMP-LNAME	Country	TEL-NO
106519	Mathew	Thomas	India	2431322
123456	Kumar	Rajeev	India	2427982
123243	Martin	Jose	India	2437981

(*c*) Changing existing data of a file

[View full size image]



Changing existing data:

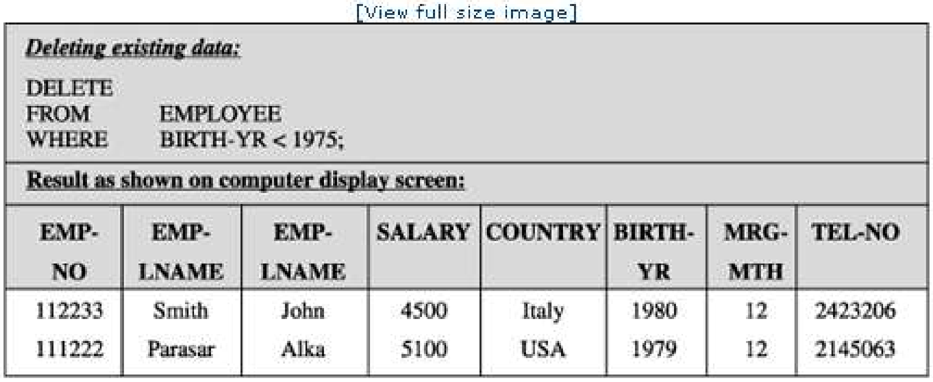
```
UPDATE     EMPLOYEE
SET        SALARY = 6000
WHERE      COUNTRY = 'INDIA';
```

Result as shown on computer display screen:

EMP-NO	EMP-LNAME	EMP-LNAME	SALARY	COUNTRY	BIRTH-YR	MRG-MTH	TEL-NO
106519	Mathew	Thomas	6000	India	1957	05	2431322
112233	Smith	John	4500	Italy	1980	12	2423206
123456	Kumar	Rajeev	6000	India	1959	06	2427982
123243	Martin	Jose	6000	India	1965	11	2437981
109876	Singh	Abhishek	4800	USA	1973	02	2147008
111222	Parasar	Alka	5100	USA	1979	12	2145063
165243	Kumar	Avinash	6500	UK	1966	01	2407841
221122	Jose	Martin	5500	AUS	1970	12	1242217

(*d*) Deleting existing data from a file

[View full size image]



Deleting existing data:

```
DELETE
FROM       EMPLOYEE
WHERE      BIRTH-YR < 1975;
```

Result as shown on computer display screen:

EMP-NO	EMP-LNAME	EMP-LNAME	SALARY	COUNTRY	BIRTH-YR	MRG-MTH	TEL-NO
112233	Smith	John	4500	Italy	1980	12	2423206
111222	Parasar	Alka	5100	USA	1979	12	2145063

1.6. Data Administrator (DA)

A data administrator (DA) is an identified individual person in the organisation who has central responsibility of controlling data. As discussed earlier, data are important assets of an organisation.

Therefore, it is important that someone at a senior level in the organisation understands these data and the organisational needs with respect to data. Thus, a DA is this senior level person in the organisation whose job is to decide what data should be stored in the database and establish policies for maintaining and dealing with that data. He decides exactly what information is to be stored in the database, identifies the entities of the interest to the organisation and the information to be recorded about those entities. A DA decides the content of the database at an abstract level. This process performed by DA is known as logical or conceptual database design. DAs are the manager and need not be a technical person, however, knowledge of information technology helps them in an overall understanding and appreciation of the system.

1.7. Database Administrator (DBA)

A database administrator (DBA) is an individual person or group of persons with an overview of one or more databases who controls the design and the use of these databases. A DBA provides the necessary technical support for implementing policy decisions of databases. Thus, a DBA is responsible for the overall control of the system at technical level and unlike a DA, he or she is an IT professional. A DBA is the central controller of the database system who oversees and manages all the resources (such as database, DBMS and related software). The DBA is responsible for authorizing access to the database, for coordinating and monitoring its use and for acquiring software and hardware resources as needed. They are accountable for security system, appropriate response time and ensuring adequate performance of the database system and providing a variety of other technical services. The database administrator is supported with a number of staff or a team of people such as system programmers and other technical assistants.

1.7.1. Functions and Responsibilities of DBAs

Following are some of the functions and responsibilities of database administrator and his staff:

a. Defining conceptual schema and database creation: A DBA creates the conceptual schema (using data definition language) corresponding to the abstract level database design made by data administrator. The DBA creates the original database schema and structure of the database. The object from the schema is used by DBMS in responding to access requests.
b. Storage structure and access-method definition: DBA decides how the data is to be represented in the stored database, the process called physical database design. Database administrator defines the storage structure (called internal schema) of the database (using data definition language) and the access method of the data from the database.
c. Granting authorisation to the users: One of the important responsibilities of a DBA is the liaising with end-users to ensure availability of required data to them. A DBA grants access to use the database to its users. It regulates the usage of specific parts of the database by various users. The authorisation information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system. DBAs assist the user with problem definition and its resolution.
d. Physical organisation modification: The DBA carries out the changes or modification to the description of the database or its relationship to the physical organisation of the database to reflect the changing needs of the organisation or to alter the physical organisation to improve performance.

e. Routine maintenance: The DBA maintains periodical back-ups of the database, either onto hard disks, compact disks or onto remote servers, to prevent loss of data in case of disasters. It ensures that enough free storage space is available for normal operations and upgrading disk space as required. A DBA is also responsible for repairing damage to the database due to misuse or software and hardware failures. DBAs define and implement an appropriate damage control mechanism involving periodic unloading or dumping of the database to backup storage device and reloading the database from the most recent dump whenever required.

f. Job monitoring: DBAs monitor jobs running on the database and ensure that performance is not degraded by very expensive tasks submitted by some users. With change in requirements (for example, reorganising of database), DBAs are responsible for making appropriate adjustment or tuning of the database

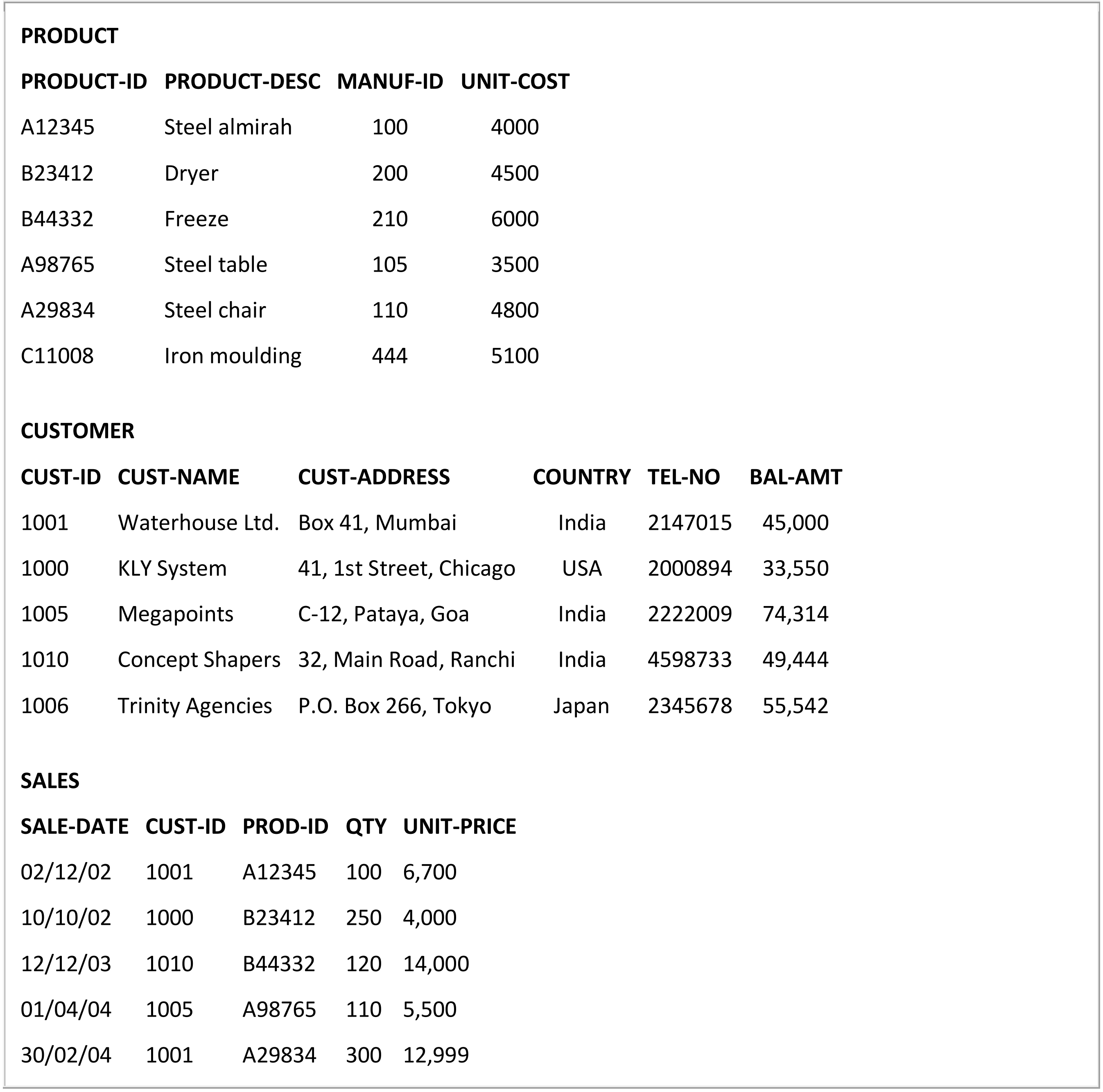
1.8. File-Oriented System versus Database System

Computer-based data processing systems were initially used for scientific and engineering calculations. With increased complexity of business requirements, gradually they were introduced into the business applications. The manual method of filing systems of an organisation, such as to hold all internal and external correspondence relating to a project or activity, client, task, product, customer or employee, was maintaining different manual folders. These files or folders were labelled and stored in one or more cabinets or almirahs under lock and key for safety and security reasons. As and when required, the concerned person in the organisation used to search for a specific folder or file serially starting from the first entry. Alternatively, files were indexed to help locate the file or folder more quickly. Ideally, the contents of each file folder were logically related. For example, a file folder in a supplier’s office might contain customer data; one file folder for each customer. All data in that folder described only that customer’s transaction. Similarly, a personnel manager might organise personnel data of employees by category of employment (for example, technical, secretarial, sales, administrative, and so on). Therefore, a file folder leveled ‘technical’ would contain data pertaining to only those people whose duties were properly classified as technical.

The manual system worked well as data repository as long as the data collection were relatively small and the organisation’s managers had few reporting requirements. However, as the organisation grew and as the reporting requirements became more complex, it became difficult in keeping track of data in the manual file system. Also, report generation from a manual file system could be slow and cumbersome. Thus, this manual filing system was replaced with a computer-based filing system. File-oriented systems were an early attempt to computerize the manual filing system that we are familiar with. Because these systems performed normal record-keeping functions, they were called data processing (DP) systems. Rather than establish a centralised store for organisation’s operational data, a decentralised approach was taken, where each department, with the assistance of DP department staff, stored and controlled its own data.

Table 1.5 shows an example of file-oriented system of an organisation engaged in product distribution. Each table represents a file in the system, for example, PRODUCT file, CUSTOMER file, SALES file and so on. Each row in these files represents a record in the file. PRODUCT file contains 6 records and each of these records contains data about different products. The individual data items or fields in the PRODUCT file are PRODUCT-ID, PRODUCT-DESC, MANUF-ID and UNIT-COST. CUSTOMER file contains 5 records and each of these records contains data about customer. The individual data items in CUSTOMER file are CUST-ID, CUST-NAME, CUST-ADDRESS, COUNTRY, TEL-NO and BAL–AMT. Similarly, SALES file contains 5 records and each of these records contains data about sales activities. The individual data items in SALES file are SALES-DATE, CUST-ID, PROD-ID, QTY and UNIT-PRICE.

Table 1.5. File-oriented system



PRODUCT

PRODUCT-ID	PRODUCT-DESC	MANUF-ID	UNIT-COST
A12345	Steel almirah	100	4000
B23412	Dryer	200	4500
B44332	Freeze	210	6000
A98765	Steel table	105	3500
A29834	Steel chair	110	4800
C11008	Iron moulding	444	5100

CUSTOMER

CUST-ID	CUST-NAME	CUST-ADDRESS	COUNTRY	TEL-NO	BAL-AMT
1001	Waterhouse Ltd.	Box 41, Mumbai	India	2147015	45,000
1000	KLY System	41, 1st Street, Chicago	USA	2000894	33,550
1005	Megapoints	C-12, Pataya, Goa	India	2222009	74,314
1010	Concept Shapers	32, Main Road, Ranchi	India	4598733	49,444
1006	Trinity Agencies	P.O. Box 266, Tokyo	Japan	2345678	55,542

SALES

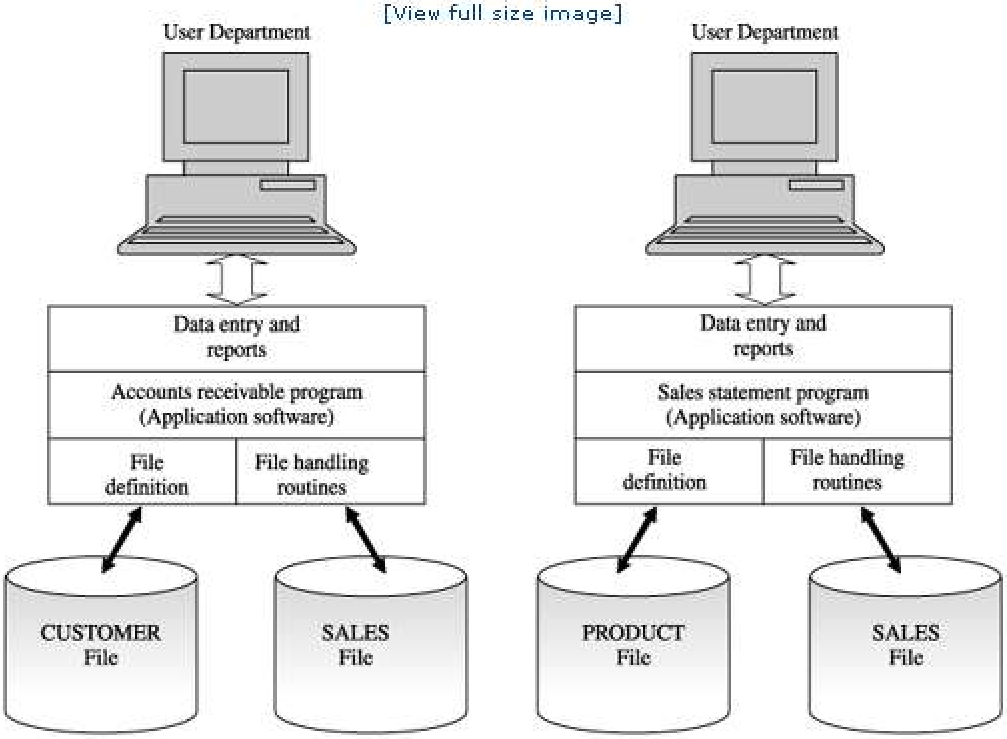
SALE-DATE	CUST-ID	PROD-ID	QTY	UNIT-PRICE
02/12/02	1001	A12345	100	6,700
10/10/02	1000	B23412	250	4,000
12/12/03	1010	B44332	120	14,000
01/04/04	1005	A98765	110	5,500
30/02/04	1001	A29834	300	12,999

With the assistance of DP department, the files were used for a number of different applications by the user departments, for example, account receivable program written to generate billing statements for customers. This program used the CUSTOMER and SALES files and these files were both stored in the computer in order by CUST-ID and were merged to create a printed statement. Similarly, sales statement generation program (using PRODUCT and SALES files) was written to generate product-wise sales performance. This type of program, which accomplishes a specific task of practical value in a business situation is called application program or application software. Each application program that is developed is designed to meet the specific needs of the particular requesting department or user group.

Fig. 1.18 illustrates structures in which application programs are written specifically for each user department for accessing their own files. Each set of departmental programs handles data entry, file maintenance and the generation of a fixed set of specific reports. Here, the physical structure and storage of the data files and records are defined in the application program. For example:

Fig. 1.18. File-oriented system

[View full size image]



a. Sales department stores details relating to sales performance, namely SALES(SALE-DATE, CUST-ID, PROD-ID, QTY, UNIT-PRICE).
b. Customer department stores details relating to customer invoice realization summary, namely CUSTOMER (CUST-ID, CUST-NAME, CUST-ADD, COUNTRY, TEL-NO, BAL-AMT).
c. Product department stores details relating to product categorization summary, namely PRODUCT (PROD-ID, PROD-DESC, MANUF-ID, UNIT-COST).

It can be seen from the above examples that there is significant amount of duplication of data storage in different departments (for example, CUST-ID and PROD-ID), which is generally true with file-oriented system.

1.8.1. Advantages of Learning File-oriented System

Although the file-oriented system is now largely obsolete, following are the several advantages of learning file-based systems:

- It provides a useful historical perspective on how we handle data.
- The characteristics of a file-based system helps in an overall understanding of design complexity of database systems.
- Understanding the problems and knowledge of limitation inherent in the file-based system helps avoid these same problems when designing database systems and thereby resulting in smooth transition.

1.8.2. Disadvantages of File-oriented System

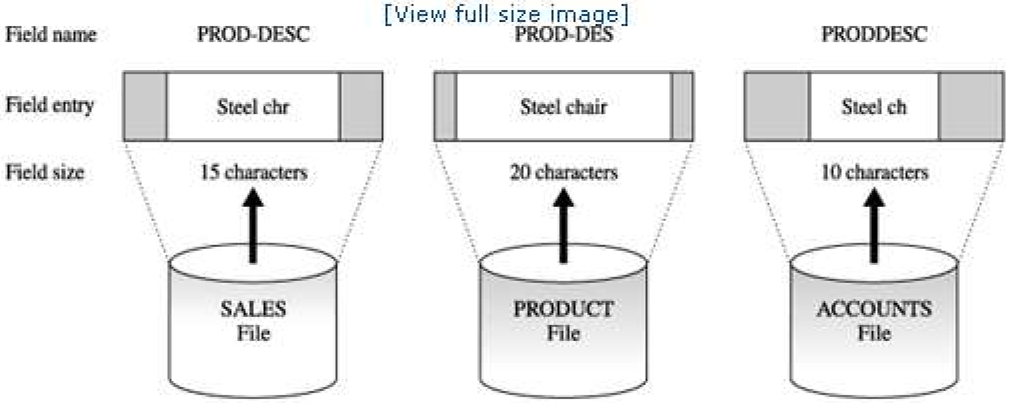
Conventional file-oriented system has the following disadvantages:

a. Data redundancy (or duplication): Since a decentralised approach was taken, each department used their own independent application programs and special files of data. This resulted into duplication of same data and information in several files, for example, duplication of PRODUCT-ID data in both PRODUCT and SALES files, and CUST-ID data in both CUSTOMER and SALES files as shown in Table 1.5. This redundancy or duplication of data is wasteful and requires additional or higher storage space, costs extra time and money, and requires increased effort to keep all files up-to- date.
b. Data inconsistency (or loss of data integrity): Data redundancy also leads to data inconsistency (or loss of data integrity), since either the data formats may be inconsistent or data values (various copies of the same data) may no longer agree or both.

   Fig. 1.19 shows an example of data inconsistency in which a field for product description is being shown by all the three department files, namely SALES, PRODUCT and ACCOUNTS. It can been seen in this example that even though it was always the product description, the related field in all the three department files often had a different name, for example, PROD-DESC, PROD-DES and PRODDESC. Also, the same data field might have different length in the various files, for example, 15 characters in SALES file, 20 characters in PRODUCT file and 10 characters in ACCOUNTS file. Furthermore, suppose a product description was changed from steel cabinet to steel chair. This duplication (or redundancy) of data increased the maintenance overhead and storage costs. As shown in Fig. 1.19, the product description filed might be immediately updated in the SALES file, updated incorrectly next week in the PRODUCT file as well as ACCOUNT file. Over a period of time, such discrepancies can cause serious degradation in the quality of information contained in the data files and can also affect the accuracy of reports.

Fig. 1.19. Inconsistent product description data

[View full size image]



- Program-data dependence: As we have seen, file descriptions (physical structure, storage of the data files and records) are defined within each application program that accesses a given file. For example, “Account receivable program” of Fig. 1.18 accesses both CUSTOMER file and SALES file. Therefore, this program contains a detailed file description for both these files. As a consequence, any change for a file structure requires changes to the file description for all programs that access the file. It can also be noticed in Fig. 1.18 that SALES file has been used in both “Account receivable program” and “Sales statement program”. If it is decided to change the CUST-ID field length from 4 characters to 6 characters, the file descriptions in each program that is affected would have to be modified to confirm to the new file structure. It is often difficult to even locate all programs affected by such changes. It could be very time consuming and subject to error when making changes. This characteristic of file-oriented system is known as program-data dependence.

- Poor data control: As shown in Fig. 1.19, a file-oriented system being decentralised in nature, there was no centralised control at the data element (field) level. It could be very common for the data field to have multiple names defined by the various departments of an organisation and depending on the file it was in. This could lead to different meanings of a data field in different context, and conversely, same meaning for different fields. This leads to a poor data control, resulting in a big confusion.

- Limited data sharing: There is limited data sharing opportunities with the traditional file-oriented system. Each application has its own private files and users have little opportunity to share data outside their own applications. To obtain data from several incompatible files in separate systems will require a major programming effort. In addition, a major management effort may also be required since different organisational units may own these different files.

- Inadequate data manipulation capabilities: Since File-oriented systems do not provide strong connections between data in different files and therefore its data manipulation capability is very limited.

- Excessive programming effort: There was a very high interdependence between program and data in file-oriented system and therefore an excessive programming effort was required for a new application program to be written. Even though an existing file may contain some of the data needed, the new application often requires a number of other data fields that may not be available in the existing file. As a result, the programmer had to rewrite the code for definitions for needed data fields from the existing file as well as definitions of all new data fields. Therefore, each new application required that the developers (or programmers) essentially start from scratch by designing new file formats and descriptions and then write the file access logic for each new program. Also, both initial and maintenance programming efforts for management information applications were significant.

- Security problems: Every user of the database system should not be allowed to access all the data. Each user should be allowed to access the data concerning his area of application only. Since, applications programs are added to the file-oriented system in an ad hoc manner, it was difficult to enforce such security system.

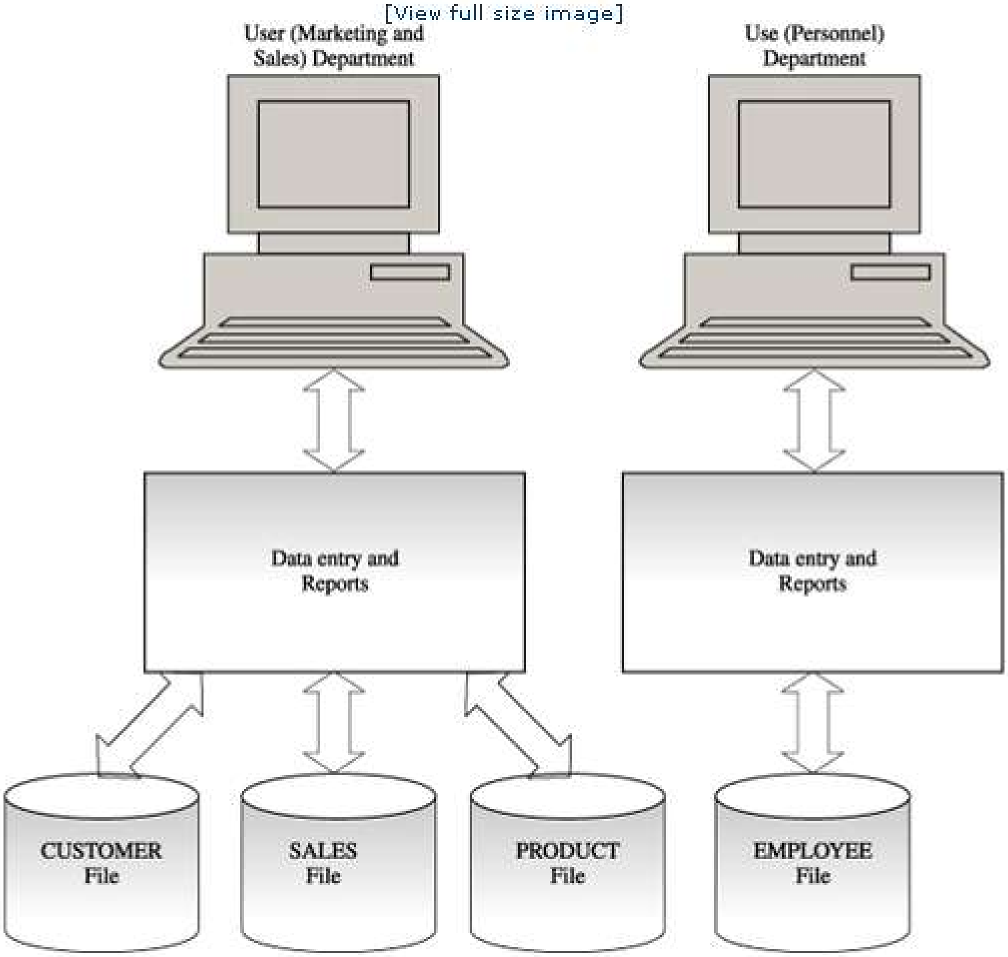
1.8.3. Database Approach

The problems inherent in file-oriented systems make using the database system very desirable. Unlike the file-oriented system, with its many separate and unrelated files, the database system consists of logically related data stored in a single data dictionary. Therefore, the database approach represents the change in the way end user data are stored, accessed and managed. It emphasizes the integration and sharing of data throughout the organisation. Database systems overcome the disadvantages of file-oriented system. They eliminate problems related with data redundancy and data control by supporting an integrated and centralised data structure. Data are controlled via a data dictionary (DD) system which itself is controlled by database administrators (DBAs). Fig. 1.20 illustrates a comparison between file-oriented and database systems.

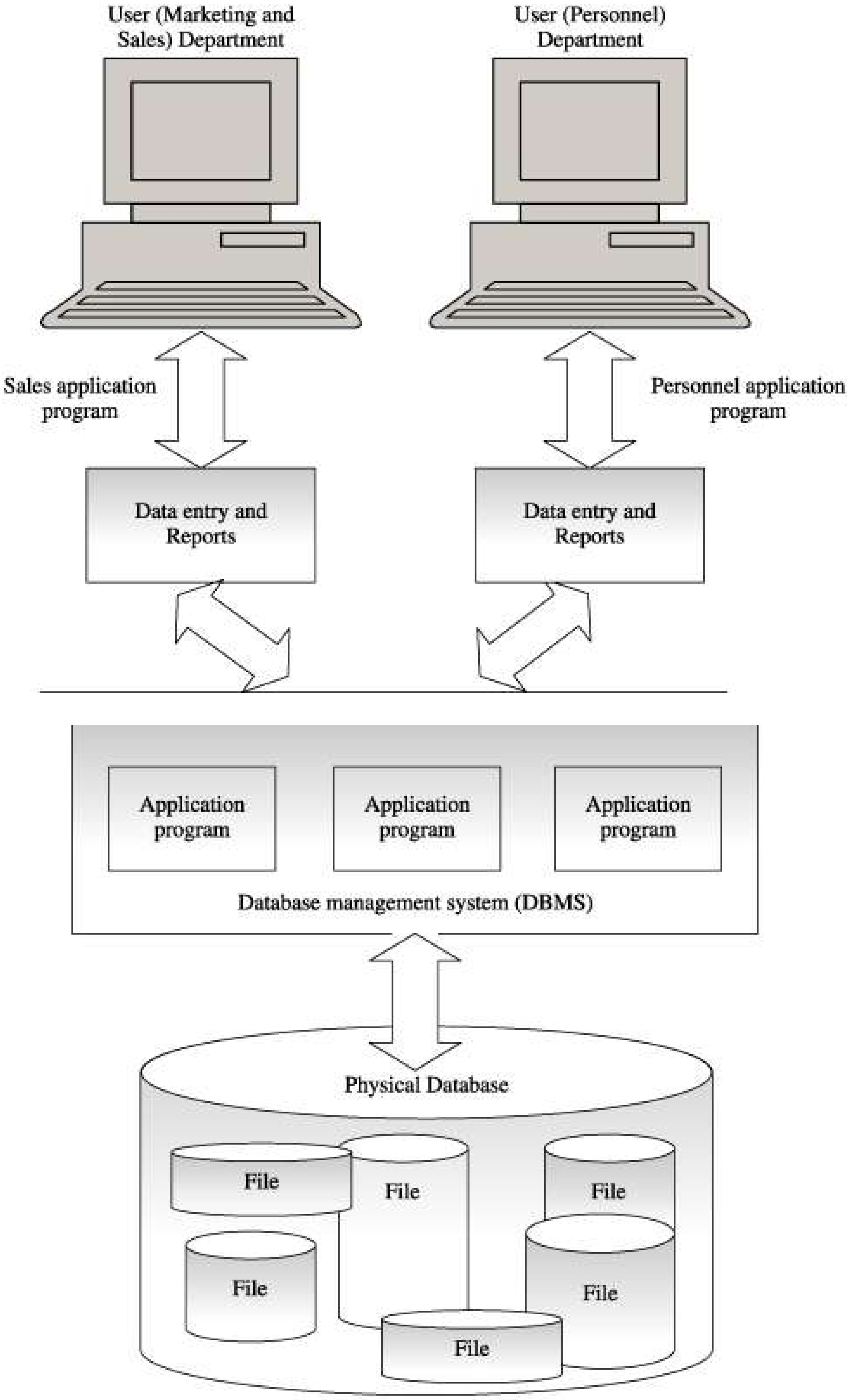
Fig. 1.20. File-oriented versus database systems

(*a*) File-oriented system

[View full size image]



(*b*) Database system


User (Marketing and Sales) Department
User (Personnel) Department
Sales application program
Personnel application program
Data entry and Reports
Data entry and Reports
Application program
Application program
Application program
Database management system (DBMS)
Physical Database
File
File
File
File
File
File

1.8.4. Database System Environment

A database system refers to an organisation of components that define and regulate the collection, storage, management and use of data within a database environment. It consists of four main parts:

- Data
- Hardware
- Software
- Users (People)

Data: From the user's point of view, the most important component of database system is perhaps the data. The term data has been explained in Section 1.2.1. The totality of data in the system is all stored in a single database, as shown in Fig. 1.20 (b). These data in a database are both integrated and shared in a system. Data integration means that the database can be thought of as a function of several otherwise distinct files, with at least partly eliminated redundancy among the files. Whereas in data sharing, individual pieces of data in the database can be shared among different users and each of those users can have access to the same piece of data, possibly for different purposes. Different users can effectively even access the same piece of data concurrently (at the same time). Such concurrent access of data by different users is possibly because of the fact that the database is integrated.

Depending on the size and requirement of an organisation or enterprise, database systems are available on machines ranging from the small personal computers to the large mainframe computers. The requirement could be a single-user system (in which at most one user can access the database at a given time) or multi-user system (in which many users can access the database at the same time).

Hardware: All the physical devices of a computer are termed as hardware. The computer can range from a personal computer (microcomputer), to a minicomputer, to a single mainframe, to a network of computers, depending upon the organisation's requirement and the size of the database. From the point of view of the database system the hardware can be divided into two components:

- The processor and associated main memory to support the execution of database system (DBMS) software and
- The secondary (or external) storage devices (for example, hard disk, magnetic disks, compact disks and so on) that are used to hold the stored data, together with the associated peripherals (for example, input/output devices, device controllers, input/output channels and so on).

A database system requires a minimum amount of main memory and disk space to run. With a large number of users, a very large amount of main memory and disk space is required to maintain and control the huge quantity of data stored in a database. In addition, high-speed computers, networks and peripherals are necessary to execute the large number of data access required to retrieve information in an acceptable amount of time. The advancement in computer hardware technology and development of powerful and less expensive computers, have resulted into increased database technology development and its application.

Software: Software is the basic interface (or layer) between the physical database and the users. It is most commonly known as database management system (DBMS). It comprises the application programs together with the operating system software. All requests from the users to access the database are handled by DBMS. DBMS provides various facilities, such as adding and deleting files, retrieving and updating data in the files and so on. Application software is generally written by company employees to solve a specific common problem.

Application programs are written typically in a third-generation programming language (3GL), such as C, C++, Visual Basic, Java, COBOL, Ada, Pascal, Fortran and so on, or using fourth-generation language (4GL), such as SQL, embedded in a third-generation language. Application programs use the facilities of the DBMS to access and manipulate data in the database, providing reports or documents needed for the information and processing needs of the organisation. The operating system software manages all hardware components and makes it possible for all other software to run on the computers.

Users: The users are the people interacting with the database system in any form. There could be various categories of users. The first category of users is the application programmers who write database application programs in some programming language. The second category of users is the end users who interact with the system from online workstations or terminals and accesses the database via one of the online application programs to get information for carrying out their primary business responsibilities. The third category of users is the database administrators (DBAs), as explained in Section 1.7, who manage the DBMS and its proper functioning. The fourth category of users is the database designers who design the database structure.

1.8.5. Advantages of DBMS

Due to the centralised management and control, the database management system (DBMS) has numerous advantages. Some of these are as follows:

a. Minimal data redundancy: In a database system, views of different user groups (data files) are integrated during database design into a single, logical, centralised structure. By having a centralised database and centralised control of data by the DBA the unnecessary duplication of data are avoided. Each primary fact is ideally recorded in only one place in the database. The total data storage requirement is effectively reduced. It also eliminates the extra processing to trace the required data in a large volume of data. Incidentally, we do not mean or suggest that all redundancy can or necessarily should be eliminated. Sometimes there are sound business and technical reasons for maintaining multiple copies of the same data, for example, to improve performance, model relationships and so on. In a database system, however, this redundancy can be carefully controlled. That is, the DBMS is aware of it, if it exists and assumes the responsibility for propagating updates and ensuring that the multiple copies are consistent.
b. Program-data independence: The separation of metadata (data description) from the application programs that use the data is called data independence. In the database environment, it allows for changes at one level of the database without affecting other levels. These changes are absorbed by the mappings between the levels. With the database approach, metadata are stored in a central location called repository. This property of data systems allows an organisation's data to change and evolve (within limits) without changing the application programs that process the data.
c. Efficient data access: DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.
d. Improved data sharing: Since, database system is a centralised repository of data belonging to the entire organisation (all departments), it can be shared by all authorized users. Existing application programs can share the data in the database. Furthermore, new application programs can be developed on the existing data in the database to share the same data and add only that data that is not currently stored, rather having to define all data requirements again. Therefore, more users and applications can share more of the data.
e. Improved data consistency: Inconsistency is the corollary to redundancy. As explained in Section 1.8.2 (b) in the file-oriented system, when the data is duplicated and the changes made at one site are not propagated to the other site, it results into inconsistency. Such database supplies incorrect or contradictory information to its users. So, if the redundancy is removed or controlled, chances of having inconsistence data is also removed and controlled. In database system, such inconsistencies are avoided to some extent by making

them known to DBMS. DMS ensures that any change made to either of the two entries in the database is automatically applied to the other one as well. This process is known as propagating updates.

f. Improved data integrity: Data integrity means that the data contained in the database is both accurate and consistent. Integrity is usually expressed in terms of constraints, which are consistency rules that the database system should not violate. For example in Table 1.5, the marriage month (MRG-MTH) in the EMPLOYEE file might be shown as 14 instead of 12. Centralised control of data in the database system ensures that adequate checks are incorporated in the DBMS to avoid such data integrity problem. For example, an integrity check for the data field marriage date (MRG-MTH) can be introduced between the range of 01 and 12. Another integrity check can be incorporated in the database to ensure that if there is reference to a certain object, that object must exit. For example, in the case of bank's automatic teller machine (ATM), a user is not allowed to transfer fund from a nonexistent saving to a checking account.

g. Improved security: Database security is the protection of database from unauthorised users. The database administrator (DBA) ensures that proper access procedure is followed, including proper authentication schemes for access to the DBMS and additional checks before permitting access to sensitive data. A DBA can define (which is enforced by DBMS) user names and passwords to identify people authorised to use the database. Different levels of security could be implemented for various types of data and operations. The access of data by authorised user may be restricted for each type of access (for example, retrieve, insert, modify, update, delete and so on) to each piece of information in the database. The enforcement of security could be data-value dependent (for example, a works manager has access to the performance details of employees in his or her department only), as well as data-type dependent (but the manager cannot access the sensitive data such as salary details of any employees, including those in his or her department).

h. Increased productivity of application development: The DBMS provides many of the standard functions that the application programmer would normally have to write in a file-oriented application. It provides all the low-level file-handling routines that are typical in application programs. The provision of these functions allows the application programmer to concentrate on the specific functionality required by the users without having to worry about low-level implementation details. DBMSs also provide a high-level (4GL) environment consisting of productivity tools, such as forms and report generators, to automate some of the activities of database design and simplify the development of database applications. This results in increased productivity of the programmer and reduced development time and cost.

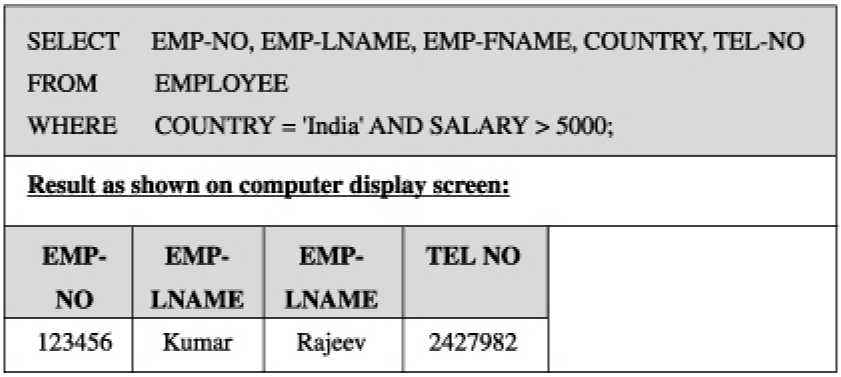
i. Enforcement of standards: With central control of the database, a DBA defines and enforces the necessary standards. Applicable standards might include any or all of the following: departmental, installation, organisational, industry, corporate, national or international. Standards can be defined for data formats to facilitate exchange of data between systems, naming conventions, display formats, report structures, terminology, documentation standards, update procedures, access rules and so on. This facilitates communication and cooperation among various departments, projects and users within the organisation. The data repository provides DBAs with a powerful set of tools for developing and enforcing these standards.

j. Economy of scale: Centralising of all the organisation's operational data into one database and creating a set of application programs that work on this source of data resulting in drastic cost savings. The DBMS approach permits consolidation of data and applications. Thus reduces the amount of wasteful overlap between activities of data-processing personnel in different projects or departments. This enables the whole organisation to invest in more powerful processors, storage devices or communication gear, rather than having each department purchase its own (low-end) equipment. Thus, a combined low cost budget is required (instead of accumulated large budget that would normally be allocated to each department for file-oriented system) for the maintenance and development of system. This reduces overall costs of operation and management, leading to an economy of scale.

k. Balance of conflicting requirements: Knowing the overall requirements of the organisation (instead of the requirements of individual users), the DBA resolves the conflicting requirements of various users and applications. A DBA can structure the system to provide an overall service that is best for the organisation. A DBA can chose the best file structure and access methods to get optimal performance for the response-critical operations, while permitting less critical applications to continue to use the database (with a relatively slower response). For example, a physical representation can be chosen for the data in storage that gives fast access for the most important applications.

l. Improved data accessibility and responsiveness: As a result of integration in database system, data that crosses departmental boundaries is directly accessible to the end-users. This provides a system with potentially much more functionality. Many DBMSs provide query languages or report writers that allow users to ask ad hoc questions and to obtain the required information almost immediately at their terminal, without requiring a programmer to write some software to extract this information from the database. For example (from Table 1.4), a works manager could list from the EMPLOYEE file, all employees belonging to India with a monthly salary greater than INR 5000 by entering the following SQL command at a terminal, as shown in Fig. 1.21.

Fig. 1.21. SQL for selected data fields



```
SELECT    EMP-NO, EMP-LNAME, EMP-FNAME, COUNTRY, TEL-NO
FROM      EMPLOYEE
WHERE     COUNTRY = 'India' AND SALARY > 5000;
```

Result as shown on computer display screen:

EMP-NO	EMP-LNAME	EMP-LNAME	TEL NO
123456	Kumar	Rajeev	2427982

m. Increased concurrency: DBMSs manage concurrent databases access and prevents the problem of loss of information or loss of integrity.

n. Reduced program maintenance: The problems of high maintenance effort required in file-oriented system, as explained in Section 1.8.2 (g), are reduced in database system. In a file-oriented environment, the descriptions of data and the logic for accessing data are built into individual application programs. As a result, changes to data formats and access methods inevitably result in the need to modify application programs. In database environment, data are more independent of the application programs.

o. Improved backup and recovery services: DBMS provides facilities for recovering from hardware or software failures through its back up and recovery subsystem. For example, if the computer system fails in the middle of a complex update program, the recovery subsystem is responsible and makes sure that the database is restored to the state it was in before the program started executing. Alternatively, the recovery subsystem ensures that the program is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

p. Improved data quality: The database system provides a number of tools and processes to improve data quality.

1.8.6. Disadvantages of DBMS

In spite of the advantages, the database approach entails some additional costs and risks that must be recognized and managed when implementing DBMS. Following are the disadvantages of using DBMS:

a. Increased complexity: A multi-user DBMS becomes an extremely complex piece of software due to expected functionality from it. It becomes necessary for database designers, developers, database administrators and end-users to understand this functionality to full advantage of it. Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organisation.

b. Requirement of new and specialized manpower: Because of rapid changes in database technology and organisation's business needs, the organisation's need to hire, train or retrain its manpower on regular basis to design and implement databases, provide database administration services and manage a staff of new people. Therefore, an organisation needs to maintain specialized skilled manpower.

c. Large size of DBMS: The large complexity and wide functionality makes the DBMS an extremely large piece of software. It occupies many gigabytes of storage disk space and requires substantial amounts of main memory to run efficiently.

d. Increased installation and management cost: The large and complex DBMS software has a high initial cost. It requires trained manpower to install and operate and also has substantial annual maintenance and support costs. Installing such a system also requires upgrades to the hardware, software and data communications systems in the organisation. Substantial training of manpower is required on an ongoing basis to keep up with new releases and upgrades. Additional or more sophisticated and costly database software may be needed to provide security and to ensure proper concurrent updating of shared data.

e. Additional hardware cost: The cost of DBMS installation varies significantly, depending on the environment and functionality, size of the hardware (for example, micro-computer, mini-computer or main-frame computer) and the recurring annual maintenance cost of hardware and software.

f. Conversion cost: The cost of conversion (both in terms of money and time) from legacy system (old file-oriented and/or older database technology) to modern DBMS environment is very high. In some situations, the cost of DBMS and extra hardware may be insignificant compared with the cost of conversion. This cost includes the cost of training manpower (staff) to use these new systems and cost of employing specialists manpower to help with the conversion and running of the system.

g. Need for explicit backup and recovery: For a centralised shared database to be accurate and available all times, a comprehensive procedure is required to be developed and used for providing backup copies of data and for restoring a database when damage occurs. A modern DBMS normally automates many more of the backup and recovery tasks than a file-oriented system.

h. Organisational conflict: A centralised and shared database (which is the case with DBMS) requires a consensus on data definitions and ownership as well as responsibilities for accurate data maintenance. As per past history and experience, sometimes there are conflicts on data definitions data formats and coding, rights to update shared data, and associated issues, which are frequent and often difficult to resolve. Organisational commitment to the database approach, organisationally astute database administrators and a sound evolutionary approach to database development is required to handle these issues.

1.10. Database Language

As explained in Section 1.5, for supporting variety of users, a DBMS must provide appropriate languages and interfaces for each category of users to express database queries and updates. Once the design of database is complete and a DBMS is chosen to implement the database, it is important to first specify the conceptual and internal schemas for the database and any mappings between the two. Following languages are used to specify database schemas:

- Data definition language (DDL)
- Storage definition language (SDL)
- View definition language (VDL)
- Data manipulation language (DML)
- Fourth-generation language (4GL)

In practice, the data definition and data manipulation languages are not two separate languages. Instead they simply form parts of a single database language and a comprehensive integrated language is used such as the widely used structured query language (SQL). SQL represents combination of DDL, VDL and DML, as well as statements for constraints specification and schema evaluation. It includes constructs for conceptual schema definition view definition, and data manipulation.

1.10.1. Data Definition Language (DDL)

Data definition (also called description) language (DDL) is a special language used to specify a database conceptual schema using set of definitions. It supports the definition or declaration of database objects (or data element). DDL allows the DBA or user to describe and name the entities, attributes and relationships required for the application, together with any associated integrity and security constraints. Theoretically, different DDLs are defined for each schema in the three-level schema-architecture (for example, for conceptual, internal and external schemas). However, in practice, there is one comprehensive DDL that allows specification of at least the conceptual and external schemas.

Various techniques are available for writing data definition language. One widely used technique is writing DDL into a text file (similar to a source program written using programming languages). Other methods use DDL compiler or interpreter to process the DDL file or statements in order to identify description of the schema constructs and to store the schema description in the DBMS catalog (or tables), which can be understood by DBMS. The result of the compilation of DDL statements is a set of tables stored in specific file collectively called the system log (explained in Section 1.2.6) or data dictionary.

For example, let us look at the following statements of DDL:

Example 1.

```
CREATE TABLE  PRODUCT
              (PROD-ID CHAR (6),
              PROD-DESC CHAR (20),
              UNIT-COST NUMERIC (4);
```

Example 2.

```
CREATE TABLE  CUSTOMER
              (CUST-ID CHAR (4),
              CUST-NAME CHAR (20),
              CUST-STREET CHAR (25),
              CUST-CITY CHAR (15)
              CUST-BAL NUMERIC (10);
```

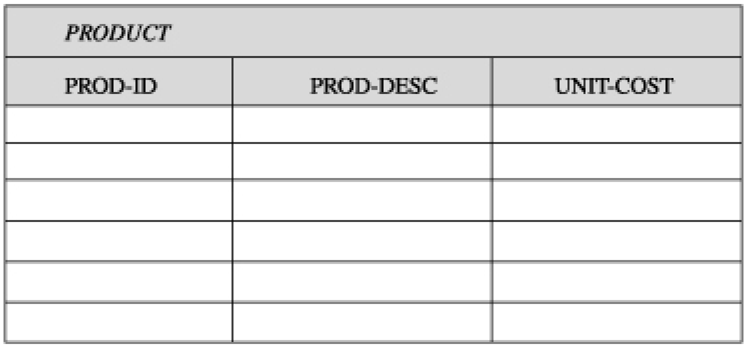
Example 3.

```
CREATE TABLE  SALES
              (CUST-ID CHAR (4),
              PROD-ID CHAR (6),
              PROD-QTY NUMERIC (3),
```

The execution of the above DDL statements will create PRODUCT, CUSTOMER and SALES tables, as illustrated in Fig. 1.23 (a), (b) and (c) respectively.

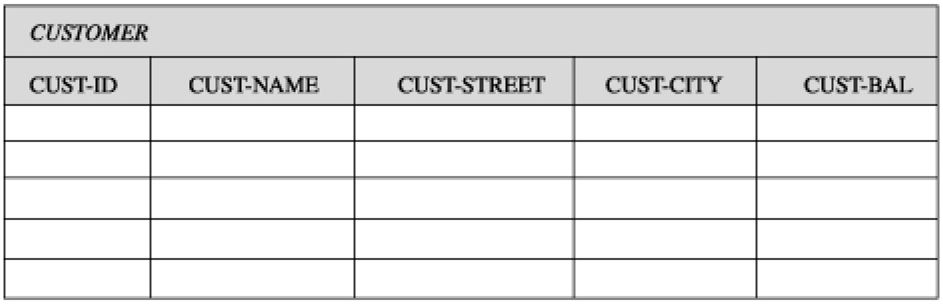
Fig. 1.23. Table creation using DDL

(*a*) Table created for PRODUCT (Example 1)



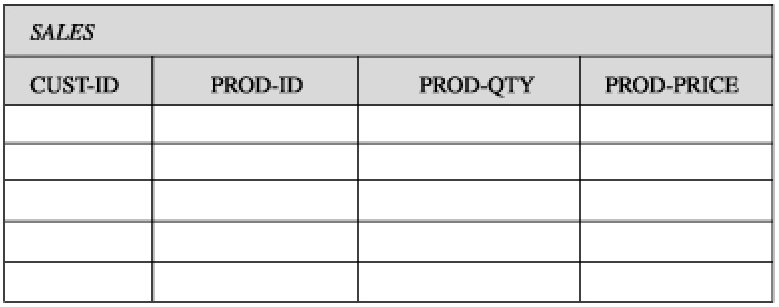
PRODUCT		
PROD-ID	PROD-DESC	UNIT-COST

(*b*) Table created for CUSTOMER (Example 2)



CUSTOMER				
CUST-ID	CUST-NAME	CUST-STREET	CUST-CITY	CUST-BAL

(*c*) Table created for SALES (Example 3)



SALES			
CUST-ID	PROD-ID	PROD-QTY	PROD-PRICE

1.10.2. Data Storage Definition Language (DSDL)

Data storage definition language (DSDL) is used to specify the internal schema in the database. The mapping between the conceptual schema (as specified by DDL) and the internal schema (as specified by DSDL) may be specified in either one of these languages. In DSDL, the storage structure and access methods used by the database system is specified by set of statements. These statements define the implementation details of the database schemas, which are usually hidden from the users.

1.10.3. View Definition Language (VDL)

View definition language (VDL) is used to specify user's views (external schema) and their mappings to the conceptual schema. However, in most of DBMSs, DDL is used to specify both conceptual and external schemas.

There are two views of data. One is the logical view of data. This is the form that the programmer perceives to be in. The other is the physical view. This reflects the way that data is actually stored on disk (or other storage devices).

1.10.4. Data Manipulation Language (DML)

Data manipulation language (DML) is a mechanism that provides a set of operations to support the basic data manipulation operations on the data held in the database. It is used to retrieve data stored in a database, express database queries and updates. In other words, it helps in communicating with DBMS. Data manipulation applies to all the three (conceptual, internal and external) levels of schema. The part of DML that provides data retrieval is called query language.

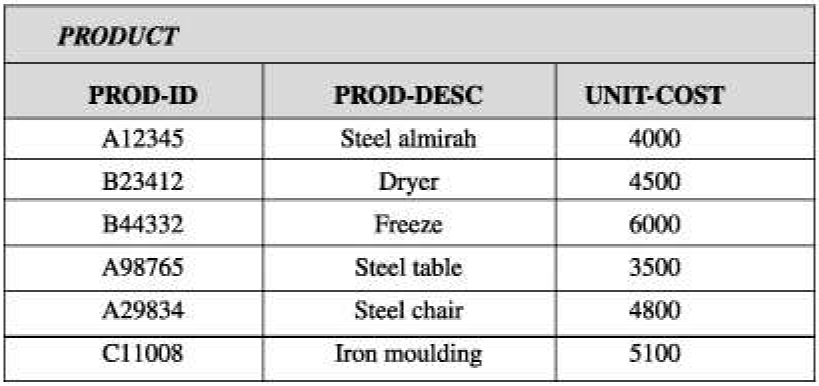
The DML provides following functional access (or manipulation operations) to the database:

- Retrieve data and/or records from database.
- Add (or insert) records to database files.
- Delete records from database files.
- Retrieve records sequentially in the key sequence.
- Retrieve records in the physically recorded sequence.
- Rewrite records that have been updated.
- Modify data and/or record in the database files.

For example, let us look at the following statements of DML that are specified to retrieve data from tables shown in Fig. 1.24.

Fig. 1.24. Retrieve data from tables using DML

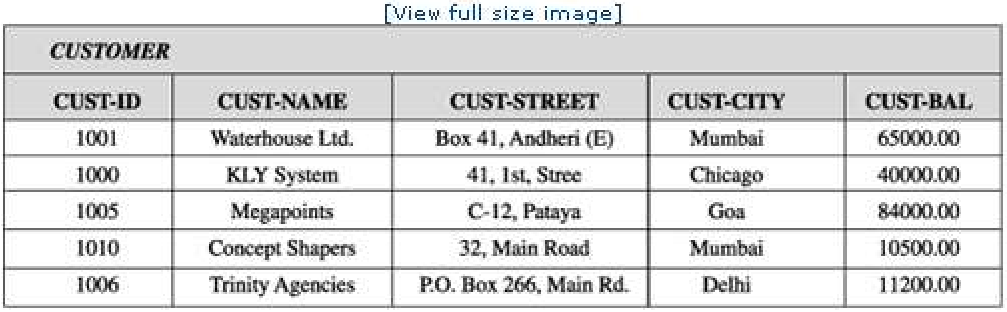
(*a*) PRODUCT table



PRODUCT		
PROD-ID	**PROD-DESC**	**UNIT-COST**
A12345	Steel almirah	4000
B23412	Dryer	4500
B44332	Freeze	6000
A98765	Steel table	3500
A29834	Steel chair	4800
C11008	Iron moulding	5100

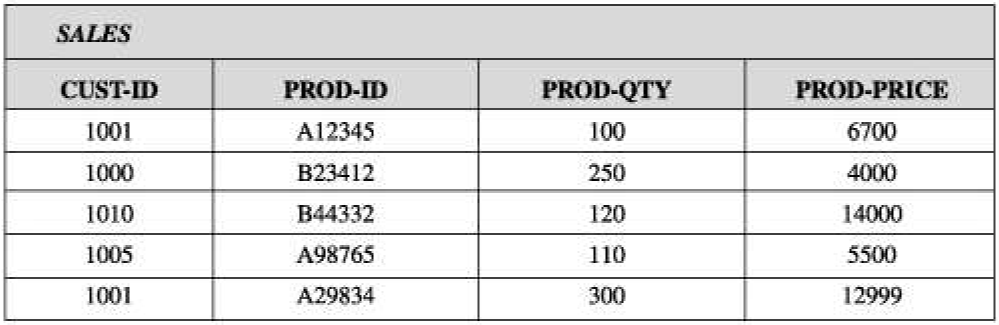
(*b*) CUSTOMER table

[View full size image]



CUSTOMER				
CUST-ID	**CUST-NAME**	**CUST-STREET**	**CUST-CITY**	**CUST-BAL**
1001	Waterhouse Ltd.	Box 41, Andheri (E)	Mumbai	65000.00
1000	KLY System	41, 1st, Stree	Chicago	40000.00
1005	Megapoints	C-12, Pataya	Goa	84000.00
1010	Concept Shapers	32, Main Road	Mumbai	10500.00
1006	Trinity Agencies	P.O. Box 266, Main Rd.	Delhi	11200.00

(*c*) SALES table



SALES			
CUST-ID	**PROD-ID**	**PROD-QTY**	**PROD-PRICE**
1001	A12345	100	6700
1000	B23412	250	4000
1010	B44332	120	14000
1005	A98765	110	5500
1001	A29834	300	12999

Example 1.

```
SELECT   PRODUCT.PROD-DESC
FROM     PRODUCT
WHERE    PROD-ID = ‘B4432’;
```

The above query (or DML statement) specifies that those rows from the table PRODUCT where the PROD-ID is B4432 should be retrieved and the PROD-DESC attribute of these rows should be displayed on the screen.

Once this query is run for table PRODUCT, as shown in Fig. 1.24 (a), the result will be displayed on the computer screen as shown below.

B44332	Freeze

Example 2.

```
SELECT   CUSTOMER.CUST-ID,
         CUSTOMER.CUST-NAME,

FROM     CUSTOMER

WHERE    CUST-CITY = ‘Mumbai’;
```

The above query (or DML statement) specifies that those rows from the table CUSTOMER where the CUST-CITY is INDIA will be retrieved. The CUST-ID, CUST-NAME and CUST-TEL attributes of these rows will be displayed on the screen.

Once this query is run for table PRODUCT, as shown in Fig. 1.24 (b), the result will be displayed on the computer screen as shown below.

1001	Waterhouse Ltd.
1010	Concept Shapers

DML query may be used for retrieving information from more than one table as explained in example 3 below.

Example 3.

```
SELECT   CUSTOMER.CUST-NAME
         CUSTOMER.CUST-BAL

FROM     SALES.PROD-ID

WHERE    SALES.PROD-ID = ‘B23412’

AND      CUSTOMER.CUST-ID = SALES.CUST-ID;
```

The above query (or DML statement) specifies that those rows from the tables CUSTOMER and SALES where the PROD-ID = B23412 and CUST-ID is same in both the tables will be retrieved and the CUST-BAL attribute of that row will be displayed on the screen.

Once this query is run for tables CUSTOMER and SALES, as shown in Fig. 1.24 (b) and (c), the result will be displayed on the computer screen as shown below.

KLY System	40000.00

There are two ways of accessing (or retrieving) data from the database. In one way, an application program issues an instruction (called embedded statements) to the DBMS to find certain data in the database and returns it to the program. This is called procedural DML. Procedural DML allows the user to tell the system what data is needed and exactly how to retrieve the data. Procedural DML retrieves a record, processes it and retrieves another record based on the results obtained by this processing and so on. The process of such retrievals continues until the data request from the retrieval has been obtained. Procedural DML is embedded in a high-level language, which contains constructs to facilitate iteration and handle navigational logic.

In the second way of accessing the data, the person seeking data sits down at a computer display terminal and issues a command in a special language (called query) directly to the DBMS to find certain data and returns it to the display screen. This is called non-procedural DML (or declarative language). Non-procedural DML allows the user to state what data are needed, rather than how they are to be retrieved.

DBMS translates a DML statement into a procedure (or set of procedures) that manipulates the required set of records. This removes the concern of the user to know how data structures are internally implemented, what algorithms are required to retrieve and how to transform the data. This provides users with a considerable degree of data independence.

1.10.5. Fourth-generation Language (4GL)

The fourth-generation language (4GL) is a compact (a short-hand type), efficient and non-procedural programming language that is used to improve the productivity of the DBMS. In 4GL, the user defines what is to be done and not how it is to be done. The 4GL depends on higher-level 4GL tools, which are used by the users to define parameters to generate an application program. The 4GL has the following components inbuilt in it:

- Query languages
- Report generators
- Spreadsheets
- Database languages
- Application generators to define operations such as insert, retrieve and update data from the database to build applications
- High-level languages to generate application program.

Structured query language (SQL) and query by example (QBE) are the examples of fourth-generation language.

1.11. Transaction Management

All work that logically represents a single unit is called transaction. The sequence of database operations that represents a logical unit of work is grouped together as a single transaction and access a database and transforms it from one state to another. A transaction can update a record, delete a record, modify a set of records and so on. When the DBMS does a ‘commit’, the changes made by transaction are made permanent. If the changes are not be made permanent, the transaction can be ‘rollback’ and the database will remain in its original state.

When updates are performed on a database, we need some way to guarantee that a set of updates will succeed all at once or not at all. Transaction ensures that all the work completes or none of it affects the database. This is necessary in order to keep the database in a consistent state. For example, a transaction might involve transferring money from a bank saving account of a person to a checking account. While this would typically involve two separate database operations. First a withdrawal from the savings account and then a deposit into the checking account. It is logically considered one unit of work. It is not acceptable to do one operation and not the other operation because that would violate integrity of the database. Thus, both withdrawal and deposit must be completed (committed) or partial transaction must be aborted (rolled-back), so that uncompleted work does not affect database.

Consider another example of a railway reservation system in which at any given instant, it is likely that several travel agents are looking for information about available seats on various trains and routes and making new reservations. When several users (travel agents) access the railway database concurrently, the DBMS must order their request carefully to avoid conflicts. For example, when one travel agent looks for a train no. 8314 on some given day and finds an empty seat, another travel agent may simultaneously be making a reservation for the same seat, thereby making the information seen by the first agent obsolete.

Through transaction management feature, database management system must protect users from the effect of system failures or crashes. DBMS ensures that all data and status is restored to a consistent state when system is restarted after a crash or failure. For example, if the travel agent asks for a reservation to be made and the DBMS has responded saying that the reservation has been made, the reservation is not lost even if the system crashes or fails. On the other hand, if the DBMS has not yet responded to the request, but is in the process of making the necessary changes to the data while the crash occurs, the partial changes are not affected in the database when the system is restored.

Transaction has, generally, following four properties, called ACID:

- Atomicity
- Consistency
- Isolation
- Durability

Atomicity means that either all the work of a transaction or none of it is applied. With atomicity property of the transaction, other operations can only access any of the rows involved in transactional access either before the transaction occurs or after the transaction is complete, but never while the transaction is partially complete. Consistency means that the transaction's work will represent a correct (or consistent) transformation of the database's state. Isolation requires that a transaction not to be influenced by changes made by other concurrently executing transactions. Durability means that the work associated with a successfully completed transaction is applied to the database and is guaranteed to survive system or media failures.

Thus, summarising above arguments, we can say that a transaction is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of ACID (that is, atomicity, consistency, isolation and durability). Transaction management plays an important role in shaping many DBMS capabilities, including concurrency control, backup and recovery and integrity enforcement. Transaction management is further discussed in greater detail in Chapter 12.

A List of Database Management Systems

This is a list of all the database management systems that I have been able to identify. If you know of any others, then please email them to me!

The systems are listed by type: relational(R), extended-relational(X), object-relational(OR), object-oriented(OO), network(N) and hierarchical(H). Note that some vendors state that their DBMS is more than one of these. In such a case the DBMS type is specified by more than one designation. For example, Centura Software states that their Velocis database is based on both the relational and network models, and in this case the designation "RN" has been specified.

For the primary market, some liberties have been taken with regard to the use of the term "Enterprise." Specifically, if a vendor does not indicate a primary market for their DBMS, then the primary market has been specified as "Enterprise."

DBMS	Vendor	Type	Primary Market
Access (Jet, MSDE)	Microsoft	R	Desktop
Adabas D	Software AG	R	Enterprise
Adaptive Server Anywhere	Sybase	R	Mobile/Embedded
Adaptive Server Enterprise	Sybase	R	Enterprise
Advantage Database Server	Extended Systems	R	Mobile/Enterprise
Datacom	Computer Associates	R	Enterprise
DB2 Everyplace	IBM	R	Mobile
Filemaker	FileMaker Inc.	R	Desktop
IDMS	Computer Associates	R	Enterprise
Ingres ii	Computer Associates	R	Enterprise
Interbase	Inprise (Borland)	R	Open Source
MySQL	Freeware	R	Open Source
NonStop SQL	Tandem	R	Enterprise
Pervasive.SQL 2000 (Btrieve)	Pervasive Software	R	Embedded
Pervasive.SQL Workgroup	Pervasive Software	R	Enterprise (Windows 32)
Progress	Progress Software	R	Mobile/Embedded
Quadbase SQL Server	Quadbase Systems, Inc.	Relational	Enterprise
R:Base	R:Base Technologies	Relational	Enterprise
Rdb	Oracle	R	Enterprise
Red Brick	Informix (Red Brick)	R	Enterprise (Data Warehousing)
SQL Server	Microsoft	R	Enterprise
SQLBase	Centura Software	R	Mobile/Embedded
SUPRA	Cincom	R	Enterprise
Teradata	NCR	R	VLDB (Data Warehousing)
YARD-SQL	YARD Software Ltd.	R	Enterprise
TimesTen	TimesTen Performance Software	R	In-Memory
Adabas	Software AG	XR	Enterprise
Model 204	Computer Corporation of America	XR	VLDB
UniData	Informix (Ardent)	XR	Enterprise
UniVerse	Informix (Ardent)	XR	Enterprise
Cache'	InterSystems	OR	Enterprise
Cloudscape	Informix	OR	Mobile/Embedded
DB2	IBM	OR	Enterprise/VLDB

Informix Dynamic Server 2000	Informix	OR	Enterprise
Informix Extended Parallel Server	Informix	OR	VLDB (Data Warehousing)
Oracle Lite	Oracle	OR	Mobile
Oracle 8I	Oracle	OR	Enterprise
PointBase Embedded	PointBase	OR	Embedded
PointBase Mobile	PointBase	OR	Mobile
PointBase Network Server	PointBase	OR	Enterprise
PostgreSQL	Freeware	OR	Open Source
UniSQL	Cincom	OR	Enterprise
Jasmine ii	Computer Associates	OO	Enterprise
Object Store	Exceleron	OO	Enterprise
Objectivity DB	Objectivity	OO	VLDB (Scientific)
POET Object Server Suite	Poet Software	OO	Enterprise
Versant	Versant Corporation	OO	Enterprise
Raima Database Manager	Centura Software	RN	Mobile/Embedded
Velocis	Centura Software	RN	Enterprise/Embedded
Db.linux	Centura Software	RNH	Open Source/Mobile/Embedded
Db.star	Centura Software	RNH	Open Source/Mobile/Embedded
IMS DB	IBM	H	Enterprise

Entity relationship model defines the conceptual view of database. It works around real world entity and association among them. At view level, ER model is considered well for designing databases.

Entity

A real-world thing either animate or inanimate that can be easily identifiable and distinguishable. For example, in a school database, student, teachers, class and course offered can be considered as entities. All entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. Entity set may contain entities with attribute sharing similar values. For example, Students set may contain all the student of a school; likewise Teachers set may contain all the teachers of school from all faculties. Entities sets need not to be disjoint.

Attributes

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, age as attributes.

There exist a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of attributes:

- **Simple attribute:**

  Simple attributes are atomic values, which cannot be divided further. For example, student's phone-number is an atomic value of 10 digits.

- **Composite attribute:**

  Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.

- **Derived attribute:**

  Derived attributes are attributes, which do not exist physical in the database, but there values are derived from other attributes presented in the database. For example, average_salary in a department should be saved in database instead it can be derived. For another example, age can be derived from data_of_birth.

- **Single-valued attribute:**

  Single valued attributes contain on single value. For example: Social_Security_Number.

- **Multi-value attribute:**

  Multi-value attribute may contain more than one values. For example, a person can have more than one phone numbers, email_addresses etc.

These attribute types can come together in a way like:

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

Entity-set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, roll_number of a student makes her/him identifiable among students.

- **Super Key:** Set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key:** Minimal super key is called candidate key that is, supers keys for which no proper subset are a superkey. An entity set may have more than one candidate key.
- **Primary Key:** This is one of the candidate key chosen by the database designer to uniquely identify the entity set.

Relationship

The association among entities is called relationship. For example, employee entity has relation works_at with department. Another example is for student who enrolls in some course. Here, Works_at and Enrolls are called relationship.

Relationship Set:

Relationship of similar type is called relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

Degree of relationship

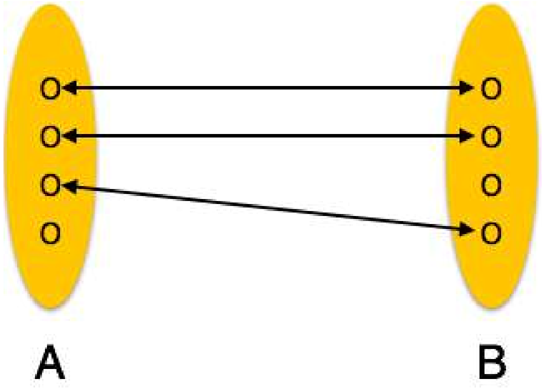
The number of participating entities in an relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

Mapping Cardinalities:

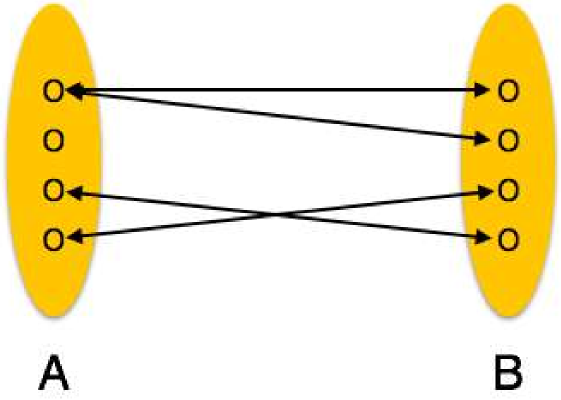
Cardinality defines the number of entities in one entity set which can be associated to the number of entities of other set via relationship set.

- **One-to-one:** one entity from entity set A can be associated with at most one entity of entity set B and vice versa.



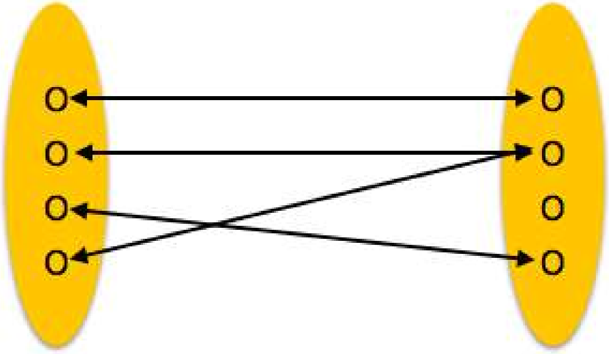
[*Image: One-to-one relation*]

One-to-many: One entity from entity set A can be associated with more than one entities of entity set B but from entity set B one entity can be associated with at most one entity.



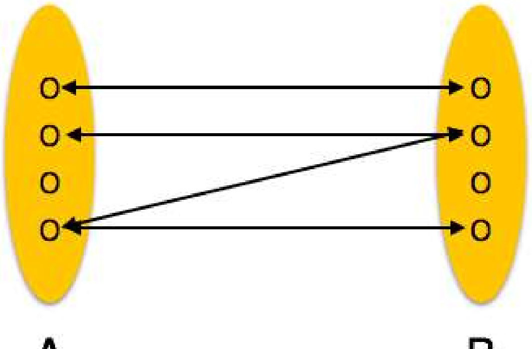
[*Image: One-to-many relation*]

Many-to-one: More than one entities from entity set A can be associated with at most one entity of entity set B but one entity from entity set B can be associated with more than one entity from entity set A.



[*Image: Many-to-one relation*]

Many-to-many: one entity from A can be associated with more than one entity from B and vice versa.



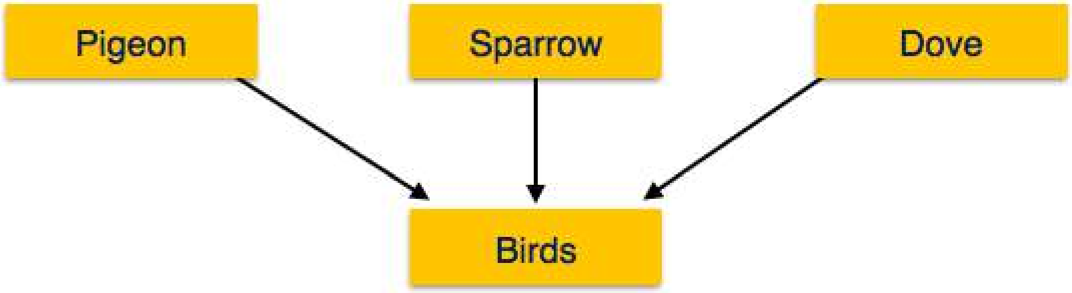
[*Image: Many-to-many relation*]

ER Model has the power of expressing database entities in conceptual hierarchical manner such that, as the hierarchical goes up it generalize the view of entities and as we go deep in the hierarchy it gives us detail of every entity included.

Going up in this structure is called generalization, where entities are clubbed together to represent a more generalized view. For example, a particular student named, Mira can be generalized along with all the students, the entity shall be student, and further a student is person. The reverse is called specialization where a person is student, and that student is Mira.

Generalization

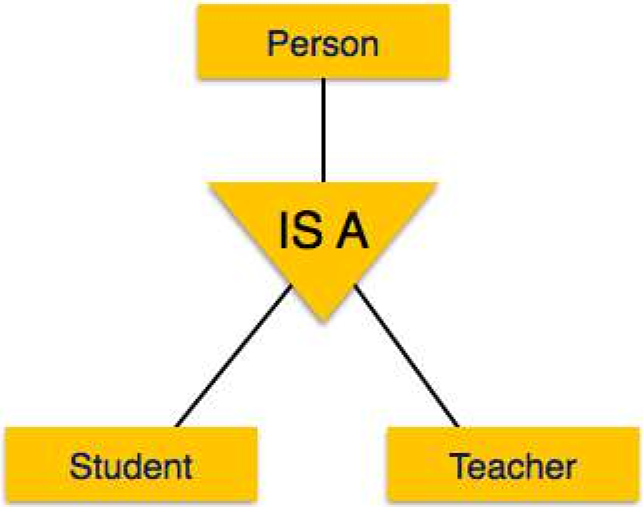
As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities is called Generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For an example, pigeon, house sparrow, crow and dove all can be generalized as Birds.



[*Image: Generalization*]

Specialization

Specialization is a process, which is opposite to generalization, as mentioned above. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group Person for example. A person has name, date of birth, gender etc. These properties are common in all persons, human beings. But in a company, a person can be identified as employee, employer, customer or vendor based on what role do they play in company.



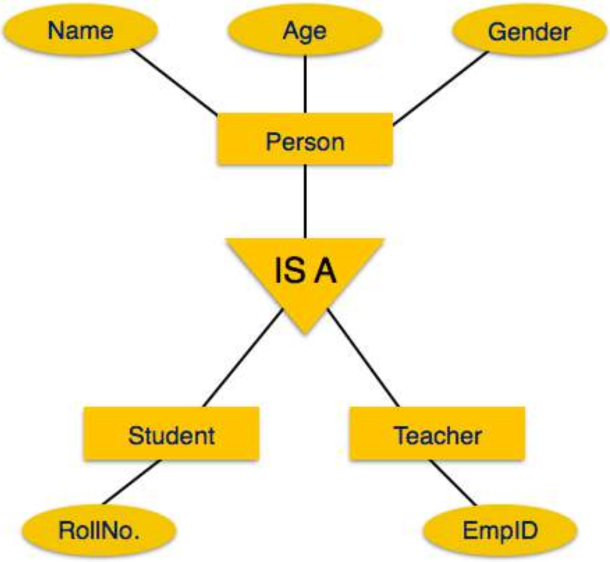
[*Image: Specialization*]

Similarly, in a school database, a person can be specialized as teacher, student or staff; based on what role do they play in school as entities.

Inheritance

We use all above features of ER-Model, in order to create classes of objects in object oriented programming. This makes it easier for the programmer to concentrate on what she is programming. Details of entities are generally hidden from the user, this process known as abstraction.

One of the important features of Generalization and Specialization, is inheritance, that is, the attributes of higher-level entities are inherited by the lower level entities.



[*Image: Inheritance*]

For example, attributes of a person like name, age, and gender can be inherited by lower level entities like student and teacher etc.

Dr Edgar F. Codd did some extensive research in Relational Model of database systems and came up with twelve rules of his own which according to him, a database must obey in order to be a true relational database.

These rules can be applied on a database system that is capable of managing is stored data using only its relational capabilities. This is a foundation rule, which provides a base to imply other rules on i**t.**