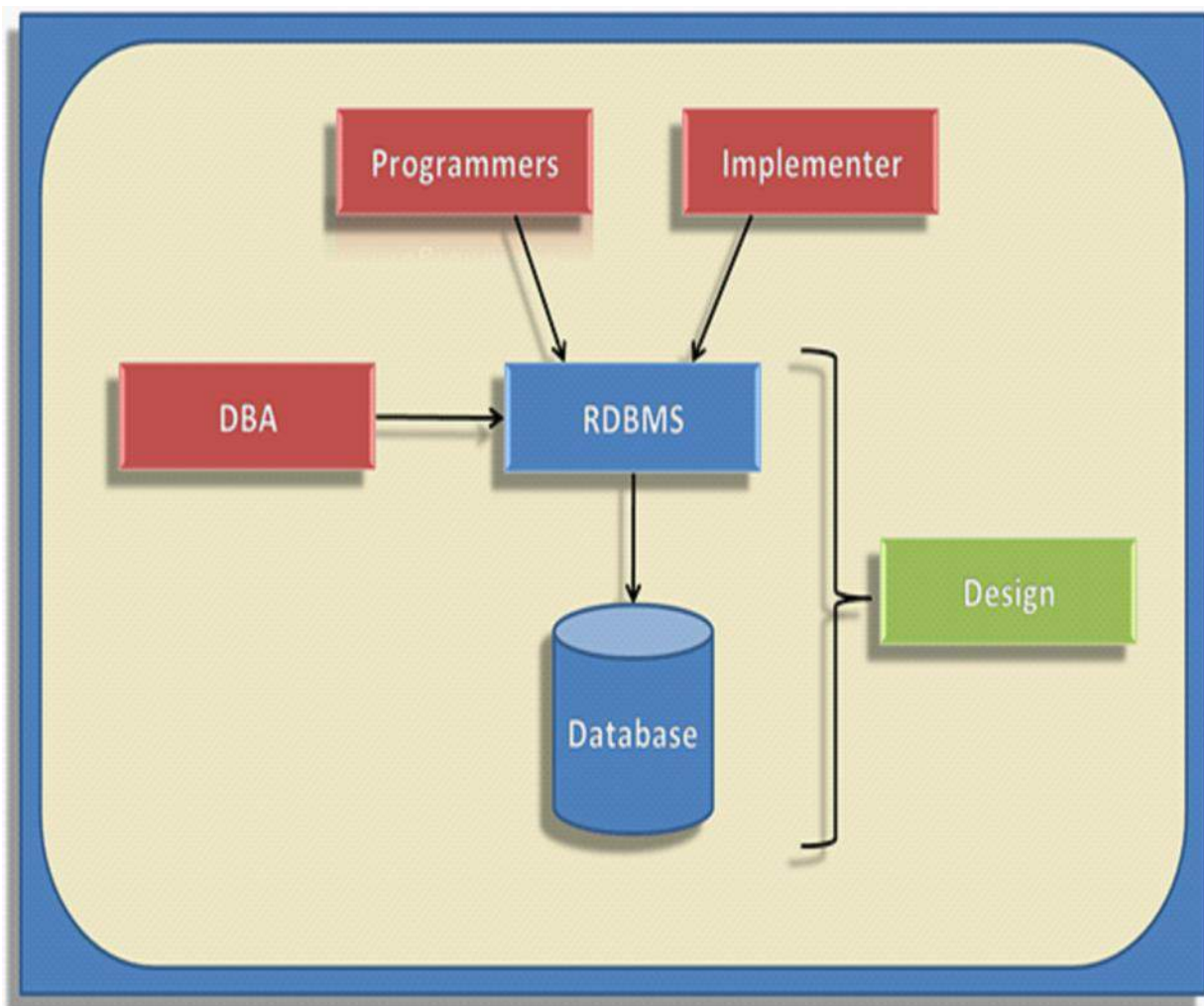


CHAPTER 13

DATABASE CONCEPTS**OBJECTIVES**

- **To understand the concept of database.**
- **Various terms used in database.**
- **To understand various data model.**
- **To understand how the keys are used in database**
- **Data warehouse and datamining.**



13.1 Introduction :

The large and complex data which are collected, entered, stored and accessed based on the users needs are in the form of queries. Unique softwares are developed and used, which are highly secured and complex. This chapter will illustrate the database concepts.

Today the database is used everywhere and in all walks of life. The database is becoming the backbone of all the softwares from standalone, client-server, on-line, mainframe, supercomputers etc. For example use in business, government agency, service organization etc. Wherever the data and information is required, there the database software is used and the results are presented in the form of reports or graphical representation, which are easier for understanding, so that future activities can be carried out based on these reports.

One of the area where database is used can be schools; for example. The Fees payment software. The basic requirements for new student admission are Name, student id, admission no, father name, class, section, amount, date of entry into the school are entered and saved. Some of the activities can be admission fees/ annual fees, monthly fees, late payment etc., created and the respective reports are generated.

DATABASE FEES SOFTWARE

MSS Fees Payment

File Edit View Help Monthly Fees Annual Fees

Date: 14-12-2014

Details:

Type: New Admission

Name: SURYA

Student ID: 450

Admission No: 450/2013-14

Fathers Name: SHIVARAMMA

Class: 10th Section: A Year: 14-2-2000

Total:

Display Clear Save And Print

| Sl No. | Particulars | Amount |
|--------|-----------------|-------------|
| 1 | Spl. Delo fee | 600 |
| 2 | Tuition fee | 1000 |
| 3 | Cap. Expn fee | 2540 |
| 4 | Extra curi. fee | 240 |
| 5 | Sports fee | 100 |
| 6 | ID Card fee | 30 |
| 7 | Appln fee | 50 |
| | Total | 4660 |

New Student

Details:

Name: SURYA

Student ID: 450

Admission No: 450/2013-14

Fathers Name: SHIVARAMMA

Class: 10th

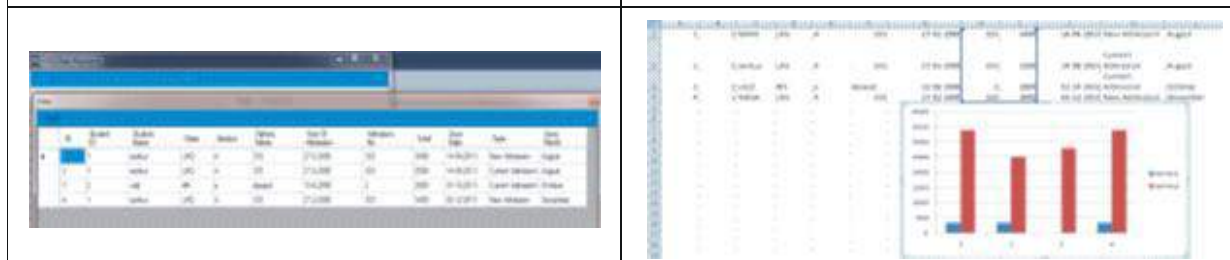
Section: A

Date: 14 Month: 2 Year: 2000

Save Clear

Reports Generated for the fees Software

Graphs and charts are generated for the Fees software



13.2 APPLICATION OF DATABASE:

Databases are widely used. Here are some representative applications:

1. **Banking:** For customer information, accounts, and loans, and banking transactions.
2. **Water meter billing :** The RR number and all the details are stored in the database and connected to the server based works.
3. **Rail and Airlines:** For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner terminals situated around the world accessed the central database system through phone lines and other data networks.
4. **Colleges :** For student information, course registrations, and grades.
5. **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
6. **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
7. **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
8. **Sales:** For customer, product, and purchase information.
9. **Manufacturing:** For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/ stores, and orders for items.
10. **Human resources:** For information about employees recruitment, salaries, payroll taxes and benefits, and for generation of paychecks.

13.3 Origin of data

The fact is something that is really occurred or is actually the case. The things that are happening and happened in real form or virtual form are considered to be fact. The fact is pursued by sense organs.

Data: Data is a collection of facts, figures, statistics, which can be processed to produce meaningful information.

The process of converting fact to data will be the first task, for any person in database concepts. The human intervention is mandatory for converting from fact to data form. The data may be in the form of letters, numbers, symbols, images, sound, video etc. The origin of fact can be from the organization/within or outside organization or any part of universe. For example the marks obtained by the student in the exam is 80, 80 is the fact, on the marks card the 80 entered will be in numeric symbols which is data.

The different forms of data and its representation is illustrated. One such form is sound represented using musical notes (software generated), these notes are stored in the form of bytes in the sound file (software digitized). In the hardware data is stored in the form of bits.

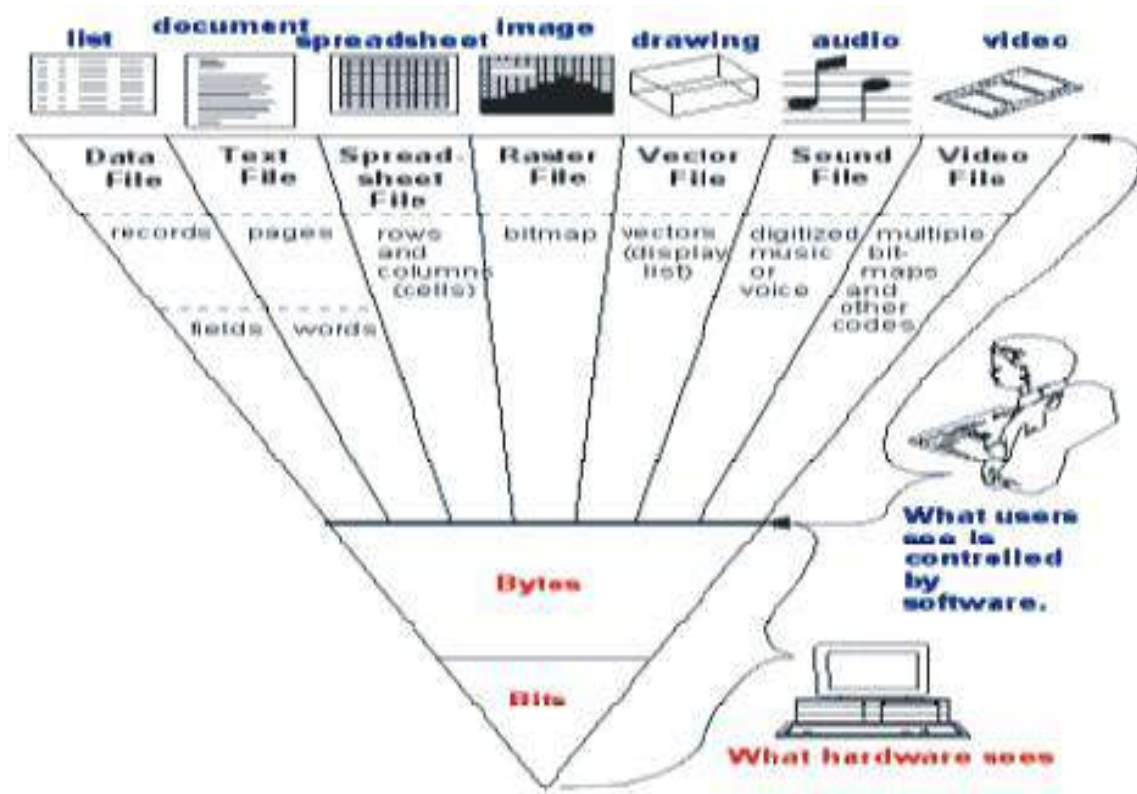


Fig. 13.1 Different forms of data

Information: Information is processed data with some definite meaning. Information represents facts, figures, or statistics, which have proper meaning.

For example, in the marks card of the student total marks, percentage and the result are processed data known as the information.

13.4 Evolution of Database - Manual File systems

Historically, When the file management came into existence, such systems were often manual, paper-and-pencil systems. The papers within these systems were organized to facilitate the expected use of the data. Typically, this was accomplished through a system of file folders and filing cabinets. As long as a collection of data was relatively small and an organization's users had few reporting requirements, the manual system served its role well as a data repository. However, as organizations grew and as reporting requirements became more complex, keeping track of data in a manual file system became more difficult. Therefore, companies looked to computer technology for help.

Computerized File systems

Initially, the computer files within the file system were similar to the manual files. The description of computer files requires a specialized vocabulary. Every discipline develops its own terminology to enable its practitioners to communicate clearly.

Differences between Manual and Computerized data processing

| Manual Data processing | Computerized Electronic Data processing |
|---|--|
| The Volume of the data, which can be processed, is limited in a desirable time. | The volume of data which can be processed can be very large. |
| Manual data processing requires large quantity of paper | Reasonable less amount of paper is used. |
| The speed and accuracy at which the job is executed is limited. | The job executed is faster and Accurate. |
| Labour cost is high. | Labour cost is economical. |
| Storage medium is paper | Storage medium is Secondary storage medium. |
| | |

13.5 DATA PROCESSING CYCLE

The way information is processed in a computer information management system. The information processing cycle consists of five specific steps:

1. **Data input**– This is any kind of data- letters, numbers, symbols, shapes, images or whatever raw material put into the computer system that needs processing. Input data is put into the computer using a keyboard, mouse or other devices such as the scanner, microphone and the digital camera. In general data must be converted to computer understandable form (English to machine code by the input devices).
2. **Data processing** – The processing is a series of actions or operations from the input data to generate outputs. some of the operations are classification based on some condition, calculation, sorting, indexing, accessing data, extracting part of filed/attribute, substring etc., conversion of data into information by the central processing unit. For example; when the computer adds $4+4=8$ that is an act of processing.
3. **Storage** - Data and information not currently being used must be stored so it can be accessed later. There are two types of storage; primary and secondary storage. Primary storage is the computer circuitry that temporarily holds data

waiting to be processed (RAM) and it is inside the computer. Secondary storage is where data is held permanently. A floppy disk, hard disk or CD- ROM is examples of this kind of storage.

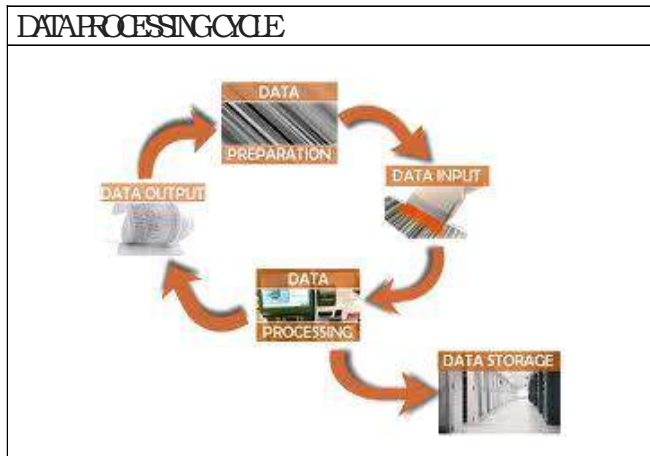


Fig. 13.2 Data Processing cycle

4 . O u t p u t – The result (information) obtained after processing the data must be presented to the user in user understandable form. The result may be in form of reports (hardcopy/ softcopy). Some of the output can be animated with sound and video/ picture.

5. Communication – Computers nowadays have communication ability which increases their power. With wired or wireless communication connections, data may be input from afar, processed in a remote area and stored in several different places and then be transmitted by modem as an e-mail or posted to the website where the online services are rendered.

13.6 Database terms :

File : File is basic unit of storage in computer system. The file is the large collection of related data.

Database: A Database is a collection of logically related data organized in a way that data can be easily accessed, managed and updated.

Tables : In Relational database, a table is a collection of data elements organized in terms of rows and columns. A table is also considered as convenient representation of relations. Table is the most simplest form of data storage. Below is an example :

Table : Employee,
Columns : Emp_Id, NAME, AGE, SALARY
Rows : There are four rows.

| Employee table | | | |
|----------------|---------|-----|----------|
| Emp_ID | NAME | AGE | SALARY |
| 1 | RAJAPPA | 43 | 45000.00 |
| 2 | ALEX | 37 | 56444.00 |
| 3 | RAMESH | 55 | 56000.00 |
| 4 | IMRAN | 28 | 60000.00 |

Fig 13.3 Table with rows and columns

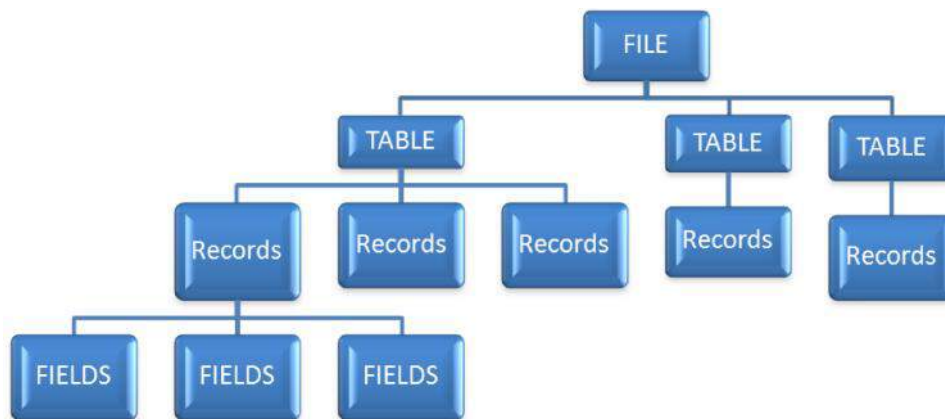


Fig 13.4 Different database terms

| | | | |
|---|------------|----|----------|
| 3 | JAIPRAKASH | 50 | 56000.00 |
|---|------------|----|----------|

Fig 13.5 Single persons details

Records: A single entry in a table is called a Record or Row. A Record in a table represents set of related data. Following is an example of single record.

Tuple :Records are also called the tuple.

Fields : Each Columns is identified by a distinct header called attribute or field

Domain :Set of values for an attribute in that column.

An Entity is an object such as a table or Form An Entity Relationship is how each table link to each other

13.7 Data types of DBMS(Data types in DBMS)

1. **Integer** – Hold whole number without fractions.
2. **Single and double precision** – Seven significant value for a number.
3. **Logical data type**-Store data that has only two values true or false.
4. **Characters** – Include letter, number, spaces, symbols and punctuation. Characters fields or variables store text information like name, address, but size will be one byte.
5. **Strings** – Sequence of character more than one. Fixed length is 0 to 63Kb and dynamic strings length range from 0 to 2 billion characters.
6. **Memo data type** – Store more than 255 characters. A memo fields can store up to 65536 characters. Long documents can store OLE objects.
7. **Index fields** –Used to store relevant information along with the documents. The document input to an index field is used to find those documents when needed. The programs provides up to 25 user definable index fields in an index set. Name drop-down look-up list, Standard, auto-complete History list.
8. **Currency fields** – The currency field accepts data in dollar form by default.
9. **Date fields** -The date fields accepts data entered in date format.
10. **Text fields** – Accepts data as an alpha-numeric text string.

13.8 DBMS-DATABASE MANAGEMENT SYSTEM

| Database | Vendor |
|------------|-----------|
| SQL Server | Microsoft |
| Oracle | Oracle |
| DB2 | IBM |
| MySQL | Oracle |
| Sybase | SAP |

A DBMS is a software that allows creation, definition and manipulation of database. DBMS is actually a tool used to perform any kind of operation on data in database. DBMS also provides protection and security to database. It maintains data consistency in case of multiple users. Here are some examples of popular DBMS, MySql, Oracle, Sybase, Microsoft Access and IBM DB2 etc.

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Features of Database System

Database approach came into existence due to the overcome of the drawbacks of file processing system. In the database approach, the data is stored at a central location and is shared among multiple users. Thus, the main advantage of DBMS

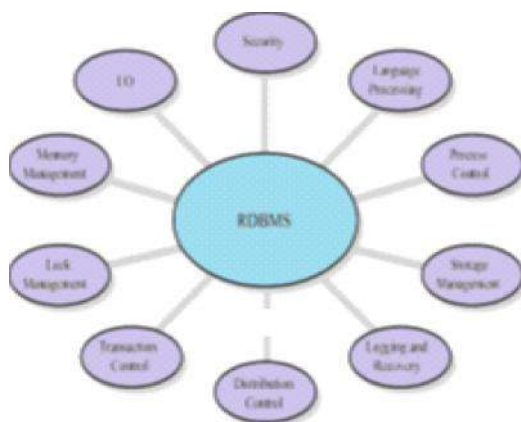


Fig 13.6 Features of DBMS

is **centralized data management**. The centralized nature of database system provides several advantages, which overcome the limitations of the conventional file processing system. These advantages are listed here.

- **Controlled data redundancy:** During database design, various files are integrated and each logical data item is stored at central location. This eliminates **replicating(duplication)** the data item in different files, and ensures consistency and saves the storage space. Note that the redundancy in the database systems cannot be eliminated completely as there could be some performance and technical reasons for having some amount of redundancy. However, the DBMS should be capable of controlling this redundancy in order to avoid data inconsistencies.

- **Enforcing data integrity:** In database approach, enforcing data integrity is much easier. Data integrity refers to the validity of data and it can be compromised

in a number of ways. Data integrity constraints can be enforced automatically by the DBMS, and others may have to be checked by the application programs.

- **Data sharing:** The data stored in the database can be shared among multiple users or application programs. Moreover, new applications can be developed to use the same stored data. Due to shared data, it is possible to satisfy the data requirements of the new applications without having to create any additional data or with minimal modification.
- **Ease of application development:** The application programmer needs to develop the application programs according to the users' needs. The other issues like concurrent access, security, data integrity, etc., are handled by the RDBMS itself. This makes the application development an easier task.
- **Data security:** Since the data is stored centrally, enforcing security constraints is much easier. The RDBMS ensures that the only means of access to the database is through an authorized channel only. Hence, data security checks can be carried out whenever access is attempted to sensitive data. To ensure security, a RDBMS provides security tools such as user codes and passwords. Different checks can be established for each type of access (addition, modification, deletion, etc.) to each piece of information in the database.
- **Multiple user interfaces:** In order to meet the needs of various users having different technical knowledge, DBMS provides different types of interfaces such as query languages, application program interfaces, and graphical user interfaces (GUI) that include forms-style and menu-driven interfaces. A form-style interface

displays a form to each user and user interacts using these forms. In menu-driven interface, the user interaction is through lists of options known as menus.

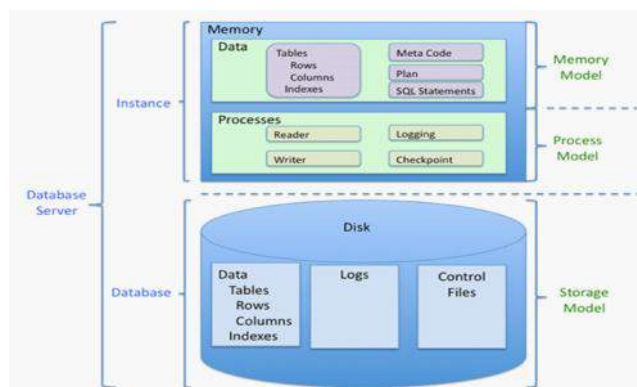


Fig 13.7 Different Layers Database

the transaction, the RDBMS recovery subsystem either reverts back the database to the state which existed prior to the start of the transaction or resumes the transaction from the point it was interrupted so that its complete effect can be recorded in the database.

- **Backup and recovery:** The RDBMS provides backup and recovery subsystem that is responsible for recovery from hardware and software failures. For example, if the failure occurs in between the

Note : Why a Spreadsheet Is Not a Database

While a spreadsheet allows for the creation of multiple tables, it does not support even the most basic data-base functionality such as support for self-documentation through metadata, enforcement of data types or domains to ensure consistency of data within a column, defined relationships among tables, or constraints to ensure consistency of data across related tables. Most users lack the necessary training to recognize the limitations of spreadsheets for these types of tasks.

13.9 Data Abstraction

The DBMS architecture describes how data in the database is viewed by the users. It is not concerned with how the data is handled and processed by the RDBMS. The database users are provided with an abstract view of the data by hiding certain details of how data is physically stored. This enables the users to manipulate the data without worrying about where it is located or how it is actually stored.

In this architecture, the overall database description can be defined at three levels, namely, internal, conceptual, and external levels and thus, named **three-level RDBMS architecture**. This architecture is proposed by ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee) and hence, is also known as ANSI/SPARC architecture.

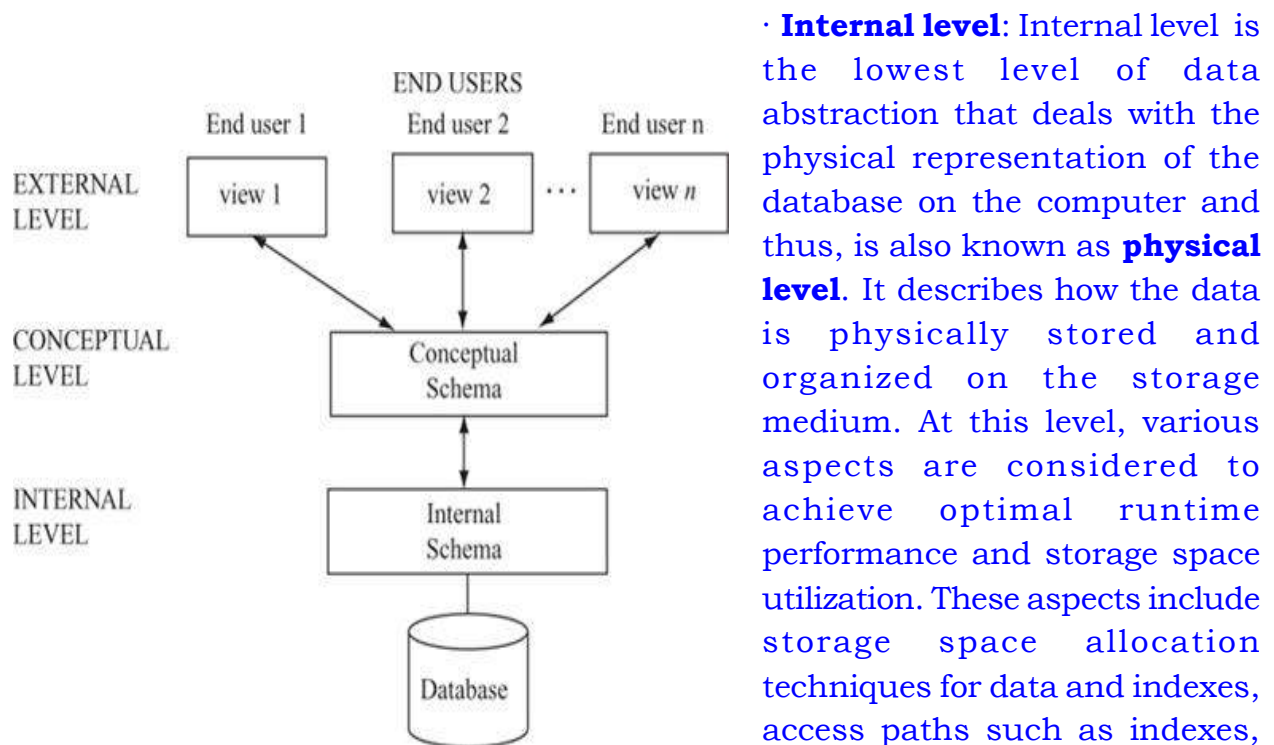


Fig 13.8 Different Levels of Database

data compression and encryption techniques, and record placement.

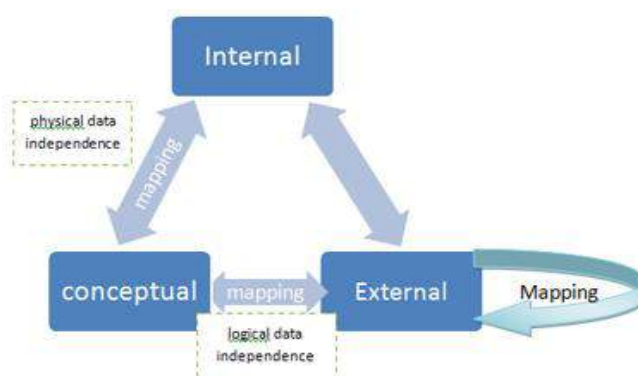
Conceptual level: This level of abstraction deals with the logical structure of the entire database and thus, is also known as **logical level**. Conceptual level describes what data is stored in the database, the relationships among the data and complete view of the user's requirements without any concern for the physical implementation. That is, it hides the complexity of physical storage structures.

The conceptual view is the overall view of the database and it includes all the information that is going to be represented in the database.

External level: External level is the highest level of abstraction that deals with the user's view of the database and thus, is also known as **view level**. In general, most of the users and application programs do not require the entire data stored in the database. The external level describes a part of the database for a particular group of users. It permits users to access data in a way that is customized according to their needs, so that the same data can be seen by different users in different ways, at the same time. In this way, it provides a powerful and flexible security mechanism by hiding the parts of the database from certain users, as the user is not aware of existence of any attributes that are missing from the view.

DBMS users : The broad classification of dbms users are

1. **Application programmers and system analysts:** System analysts determine the requirement of end users; especially naive, parametric end users, and develop specifications for transactions that meet these requirements. Application programmers implement these parameters in programs.
2. **End users :** People who require access to the database for querying updating and generating reports. The database exists primarily for/their use.
3. **Database Administrator (DBA):** DBA is responsible for authorization access to the database for coordinating and monitoring its use, and for acquiring the needed software and hardware resources.



4. **Database designers:** Database designers are responsible for identifying the data to be stored in the database for choosing appropriate structures to represent and store the data.

Schema objects are database objects that contain data or govern or perform operations on data.

13.10 Data Independence:

Data Independence is an ability of a database to modify a schema definition at one level without affecting a schema in the next higher level (the ability to change the conceptual schema without affecting the external schemas or application programs). Data independence occurs because when the schema is changed at one level, the schema at next level remains unchanged and only the mapping between the two levels is changed. Two types of data independence are:

1. Physical Data Independence.
2. Logical Data Independence

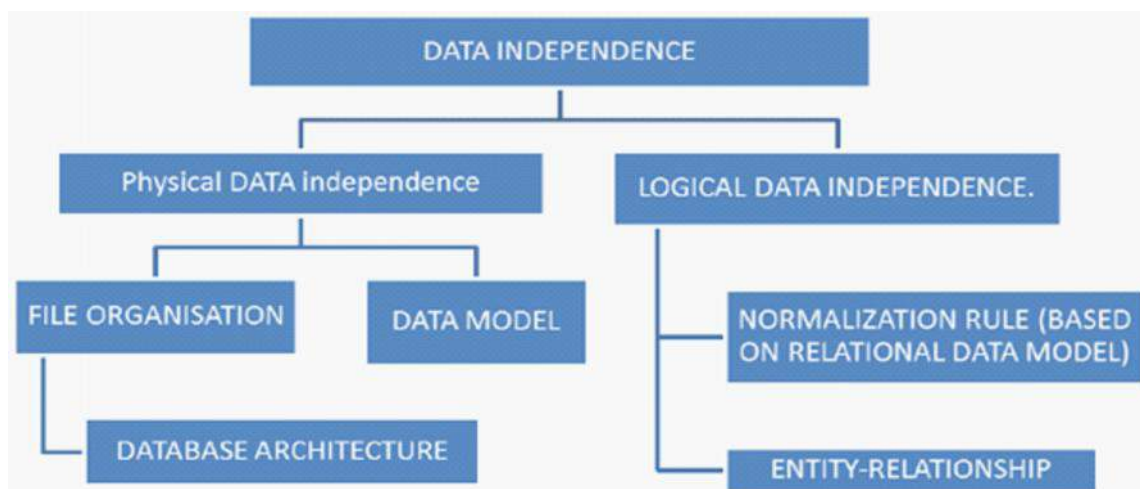


Fig 13.9 Data Independence

Physical Data independence

All schemas are logical and actual data is stored in bit format on the disk. Namely storage medium: Hard disk (all the files will be stored), floppies, drum, tapes, SD etc., System designs choose to organize, access and process records and files in different ways depending on the type of application and the needs of users. The three commonly used file organizations used in dbms/rdbms data processing applications are sequential, direct and indexed sequential access method (ISAM). The selection of a particular file organization depends upon the application used. To access a record some key field or unique identifying value that is found in every record in a file is used.

Serial File Organization : With serial file organization, records are arranged one after another, in no particular order other than the chronological order in which records are added to the file. Serial organization is commonly found in the transaction data. Where records are created in a file in the order in which transaction takes place. Serial file organization provides advantages like fast

access to next records in sequence, stored in economical storage media and easy to do the file backup facility, updating is slowly in this file organization.

Sequential File organization : Records are stored one after another in an ascending or descending order determined by the key field of the records. Example payroll file, records are stored in the form of employee id. Sequentially organized files that are processed by computer systems are normally stored on storage media such as magnetic tape, punched cards, or magnetic disks. To access these records the computer must read the file in sequence from the beginning. The first record is read and processed first, then the second record in the file sequence, and so on. To locate a particular record, the computer program must read in each record in sequence and compare its key field to the one that is needed. The retrieval search ends only when the desired key matches with the key field of the currently read record. On an average, about half of the file has to be searched to retrieve the desired record from a sequential file.

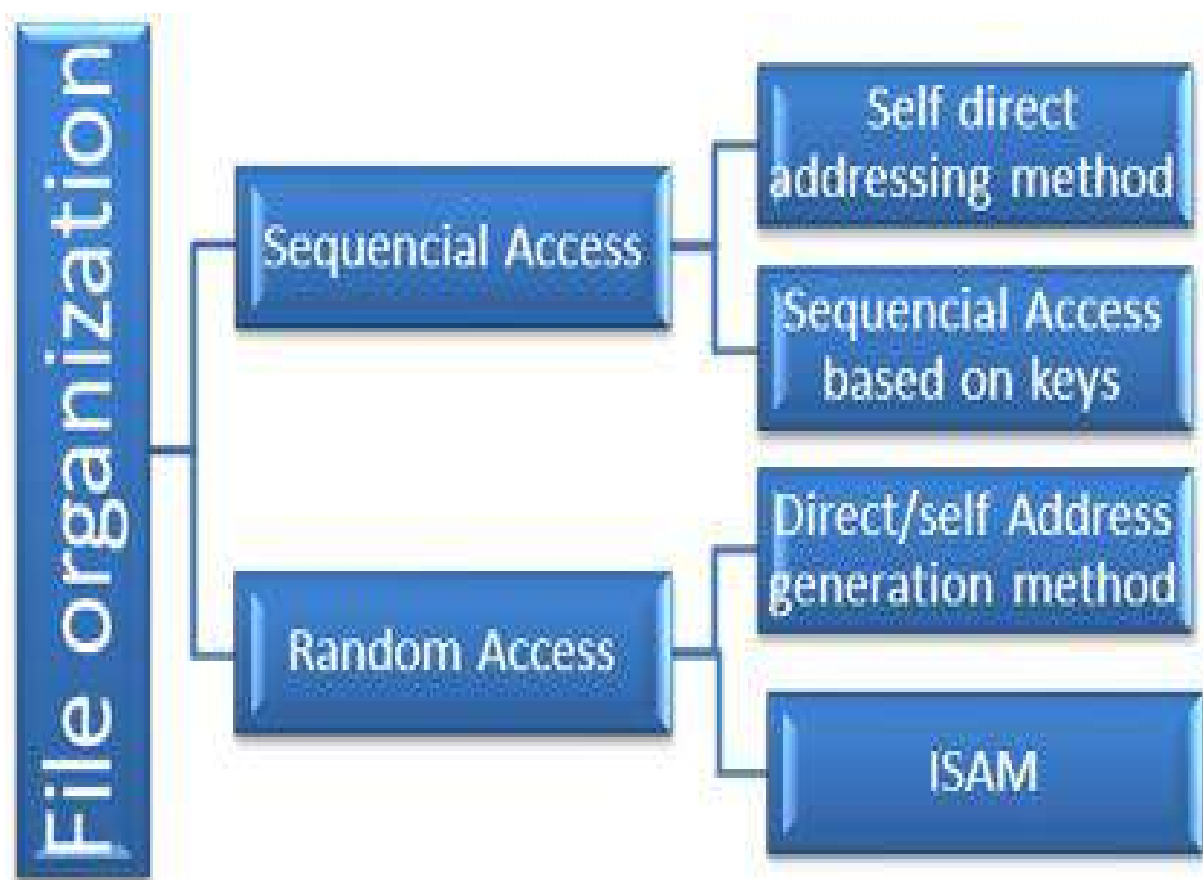


Fig 13.10 File organisation

Random/Direct Access File Organization: Direct access file organization allow immediate direct access to individual records on the file. The record are stored and retrieved using a relative record number, which gives the position of the record in the file. This type of organization also allows the file to accessed sequentially. The primary storage in a CPU truly provides for direct access. There are some devices outside the CPU which can provide the direct feature; the direct access storage devices have the capability of directly reaching any location. Although there are several types of storage devices including discs and other mass storage.

Self(direct) Addressing: Under self-direct addressing, a record key is used as its relative address. Therefore, anyone can compute the record's address from the record key and the physical address of the first record in the file. Advantage is self-addressing no need to store an index. Disadvantages are, the records must be of fixed length, if some records are deleted the space remains empty.

Random access method : Records are stored on disk by using a hasing algorithm. The key field is fed through hashing algorithm and a relative address is created. This address gives the position on the disk where the record is to be stored. The desired records can be directly accessed using randomizing procedure or hashing without accessing all other records in the file. Randomizing procedure is characterized by the fact that records are stored in such a way that there is no relationship between the keys of the adjacent records. The technique provide for converting the records key number to a physical location represented by a disk address through a computational procedure.

Advantages : The access to, and retrieval of a records is quick and direct. Transactions need not be stored and placed in sequence prior to processing Best used for online transaction.

Disadvantages: Address generation overhead is involved for accessing each record due to hashing function.

May be less efficient in the use of storage space than sequentially organized files.

Indexed Sequential Access Method(ISAM): ISAM is the hybrid between sequential and direct access file organization. The records within the file are stored sequentially but direct access to individual records is possible through an index. Indexing permit access to selected records without searching the entire file.

Advantages: ISAM permits efficient and economical use of sequential processing techniques when the activity ratio is high.

Permits direct access processing of records in a relatively efficient way when the activity ratio is low.

| Cylinder | Highest Record key in the cylinder | Cylinder 1 track index | | Cylinder 2 track index | |
|----------|------------------------------------|------------------------|------------------------------------|------------------------|------------------------------------|
| | | Track | Highest Record key in the cylinder | Track | Highest Record key in the cylinder |
| 1 | 84 | 1 | 84 | 1 | 95 |
| 2 | 250 | 2 | 250 | 2 | 110 |
| 3 | 398 | 3 | 398 | 3 | 175 |
| 4 | 479 | 4 | 479 | 4 | 250 |

Fig 13.11 Table describing ISAM

Disadvantages: Files must be stored in a direct-access storage device. Hence relatively expensive hardware and software resources are required.

Access to records may be slower than direct file.

Less efficient in the use of storage space than some other alternatives.

Different types of Architecture

The design of a Database Management System highly depends on its architecture. It can be centralized or decentralized or hierarchical. DBMS architecture can be seen as single tier or multi-tier. N-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed or replaced.

Database architecture is logically divided into three types.

1. Logical one-tier **In 1-tier architecture**,
2. Logical two-tier Client / Server architecture
3. Logical three-tier Client/Server architecture

Logical one-tier **In 1-tier architecture**

In 1-tier architecture, DBMS is the only entity where user directly sits on DBMS and uses it. Any changes done here will directly be done on DBMS itself. It does not provide handy tools for end users and preferably database designer and programmers use single tier architecture.

Fig 13.12 Logical one tier architecture



Logical two-tier Client / Server architecture**Two-tier Client / Server Architecture**

Two-tier Client / Server architecture is used for User Interface program and Application Programs that runs on client side. An interface called ODBC (Open Database Connectivity) provides an API that allows client side program to call the DBMS. Most DBMS vendors provide ODBC drivers. A client program may connect to several DBMS's. In this architecture some variation of client is also possible for example in some DBMS's more functionality is transferred to the client including data dictionary, optimization etc. Such clients are called Data server.

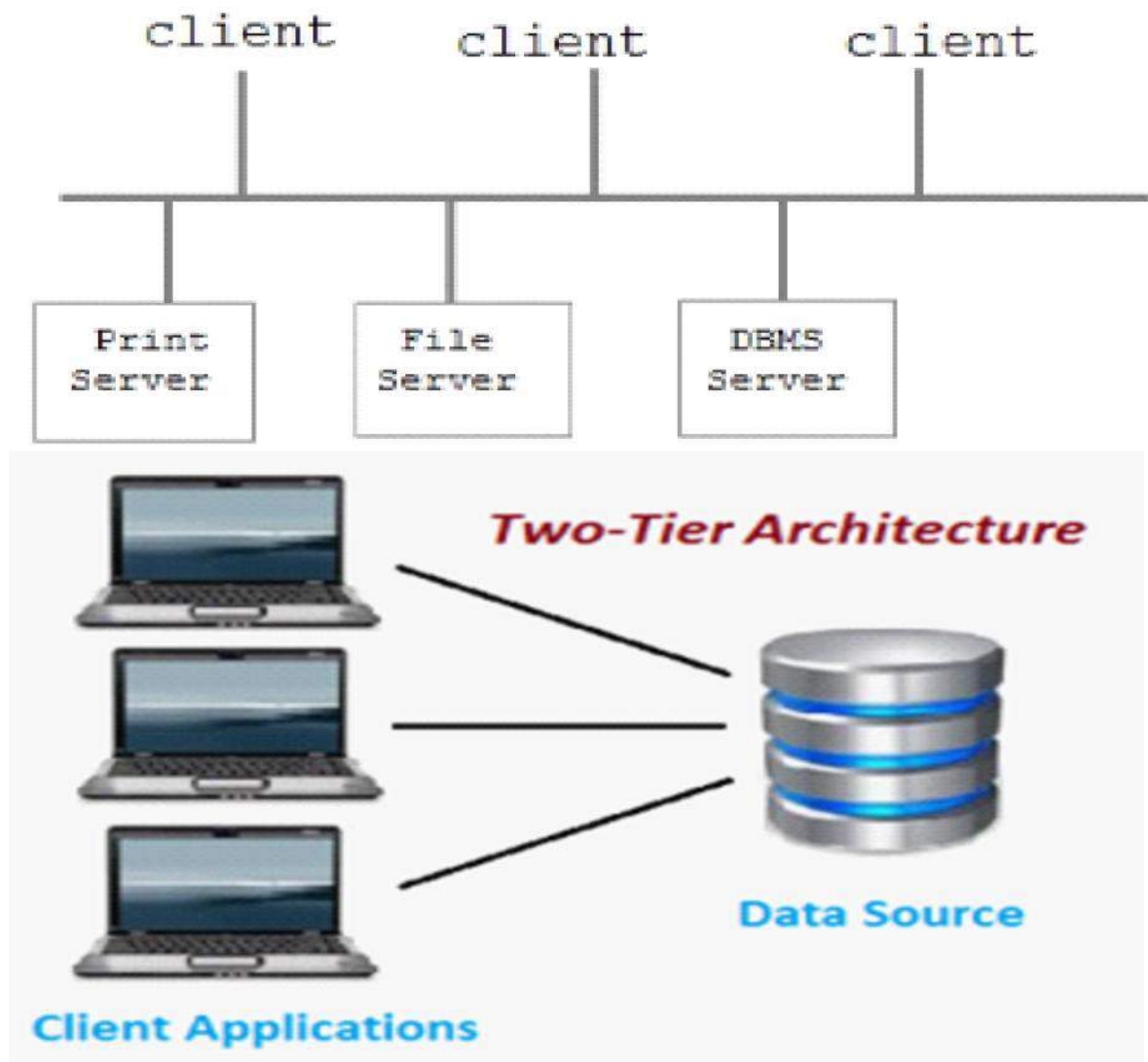


Fig 13.13 2-Level Tier Architecture

Three-tier Client / Server Architecture

Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called Application server or Web Server stores the web connectivity software and the business logic (constraints) part of application used to access the right amount of data from the database server. This layer acts like medium for sending partially processed data between the database server and the client.

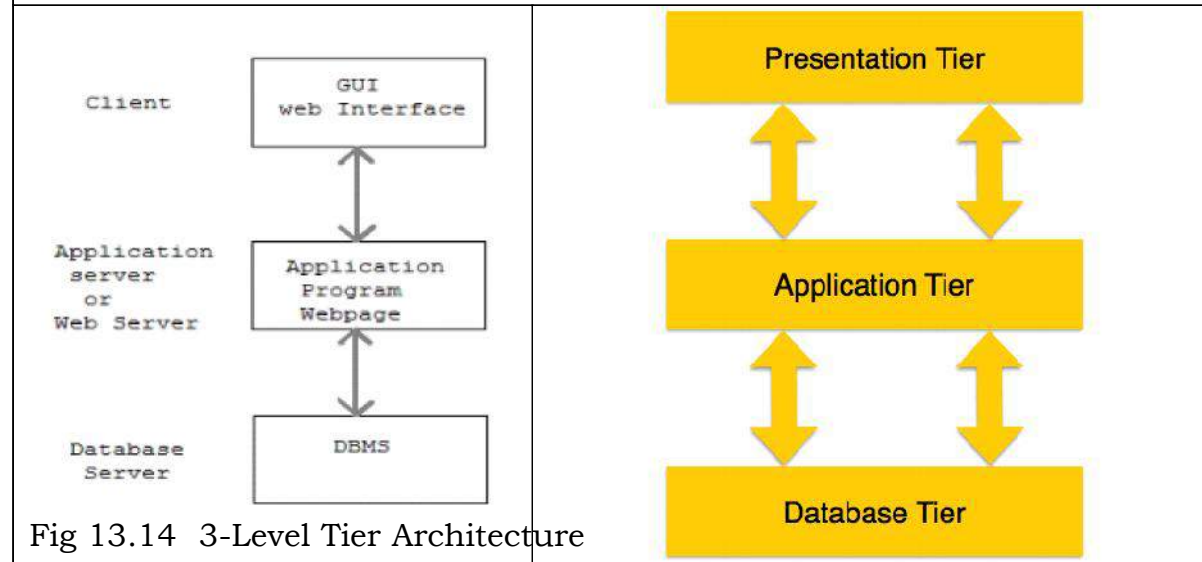


Fig 13.14 3-Level Tier Architecture

Database (Data) Tier: At this tier, only database resides. Database along with its query processing languages sits in layer-3 of 3-tier architecture. It also contains all relations and their constraints.

Application (Middle) Tier: At this tier the application server and program, which access database, resides. For a user this application tier works as abstracted view of database. Users are unaware of any existence of database beyond application. For database-tier, application tier is the user of it. Database tier is not aware of any other user beyond application tier. This tier works as mediator between the two.

User (Presentation) Tier: An end user sits on this tier. From a users aspect this tier is everything. He/she doesn't know about any existence or form of database beyond this layer. At this layer multiple views of database can be provided by the application. All views are generated by applications, which reside in application tier. Multiple tier database architecture is highly modifiable as almost all its components are independent and can be changed independently.

13.11 Database Model

A database model or simply a **data model** is an abstract model that describes how the data is represented and used. A data model consists of a set of data

structures and conceptual tools that is used to describe the structure (data types, relationships, and constraints) of a database.

A data model not only describes the structure of the data, it also defines a set of operations that can be performed on the data. A data model generally consists of **data model theory**, which is a formal description of how data may be structured and used, and **data model instance**, which is a practical data model designed for a particular application. The process of applying a data model theory to create a data model instance is known as **data modeling**.

The main objective of database system is to highlight only the essential features and to hide the storage and data organization details from the user. This is known as data abstraction. A database model provides the necessary means to achieve data abstraction.

A Database model defines the logical design of data. The model describes the relationships between different parts of the data.

In history of database design, three models have been in use.

- * Hierarchical Model
- * Network Model
- * Relational Model

13.11.1 Hierarchical Model

The hierarchical data model is the oldest type of data model, developed by IBM in 1968. This data model organizes the data in a tree-like structure, in which each **child node** (also known as **dependents**) can have only one **parent node**. The database based on the hierarchical data model comprises a set of records connected to one another through links. The link is an association between two or more records. The top of the tree structure consists of a single node that does not have any parent and is called the **root node**.

The root may have any number of dependents; each of these dependents may have any number of lower level dependents. Each child node can have only one parent node and a parent node can have any number of (many) child nodes. It, therefore, represents only one-to-one and one-to-many relationships. The collection of same type of records is known as a **record type**.

For simplicity, only few fields of each record type are shown. One complete record of each record type represents a **node**.

In this model each entity has only one parent but can have several children . At the top of hierarchy there is only one entity which is called Root.

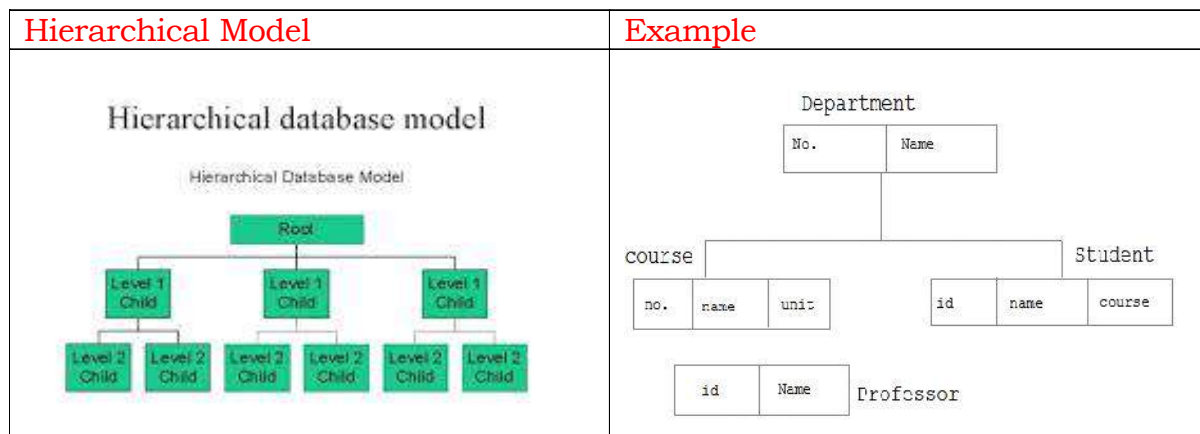


Fig 13.15 Hierarchical Model

| Advantage | Dis-advantage |
|--|---|
| <p>The hierarchical data model is that the data access is quite predictable in the structure and, therefore, both the retrieval and updates can be highly optimized by the DBMS.</p> | <p>The main drawback of this model is that the links are 'hard coded' into the data structure, that is, the link is permanently established and cannot be modified. The hard coding makes the hierarchical model rigid. In addition, the physical links make it difficult to expand or modify the database and the changes require substantial redesigning efforts.</p> |

13.11.2 Network Model

The first specification of network data model was presented by Conference on Data Systems Languages (CODASYL) in 1969, followed by the second specification in 1971. It is powerful but complicated. In a network model the data is also represented by a collection of records, and relationships among data are represented by links. However, the link in a network data model represents an association between precisely two records. Like hierarchical data model, each record of a particular record type represents a node. However, unlike hierarchical data model, all the nodes are linked to each other without any hierarchy. The main difference between hierarchical and network data model is that in

hierarchical data model, the data is organized in the form of trees and in network data model, the data is organized in the form of graphs.

In the network model, entities are organized in a graph, in which some entities can be accessed through several path

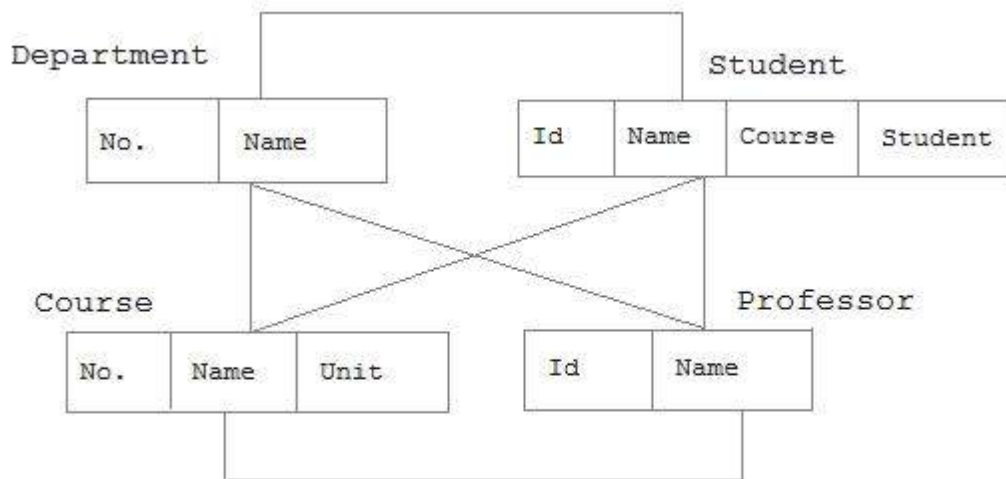


Fig 13.16 Network Model of database

Network Model of database

| Advantage | Dis-advantage |
|---|--|
| The network data model is that a parent node can have many child nodes and a child can also have many parent nodes. Thus, the network model permits the modeling of many-to-many relationships in data. | The network data model is that it can be quite complicated to maintain all the links and a single broken link can lead to problems in the database. In addition, since there are no restrictions on the number of relationships, the database design can become complex. |

13.11.3 Relational Model

The relational data model was developed by E. F. Codd in 1970. In the relational data model, unlike the hierarchical and network models, there are no physical links. All data is maintained in the form of tables (generally, known as **relations**) consisting of rows and columns. Each row (record) represents an entity and a column (field) represents an attribute of the entity. The relationship between the two tables is implemented through a common attribute in the tables and not by physical links or pointers. This makes the querying much easier in a relational database system than in the hierarchical or network database systems. Thus,

the relational model has become more programmer friendly and much more dominant and popular in both industrial and academic scenarios. Oracle, Sybase, DB2, Ingres, Informix, MS-SQL Server are few of the popular relational DBMSs.

In this model, data is organized in two-dimensional tables called relations. The tables or relation are related to each other.

Relational Model of database

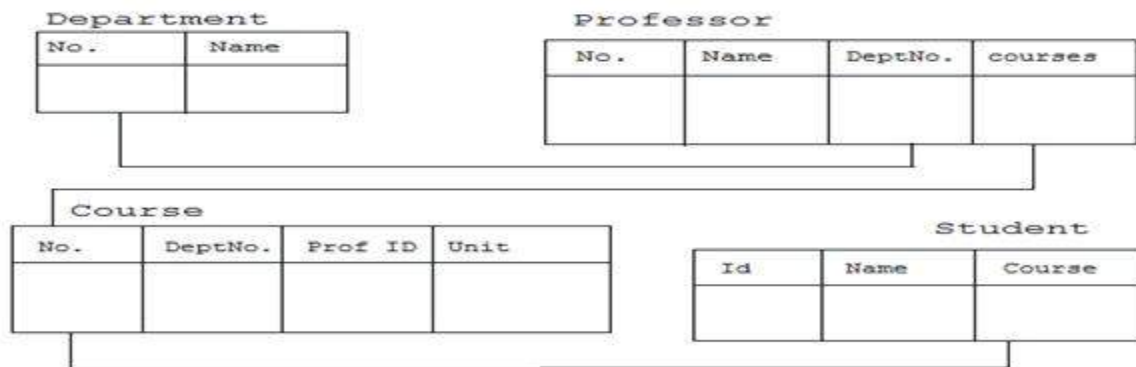


Fig 13.17 Relational Model of database

Basic Rules for the Relational Datamodel

13.12 Codd's Rule

E.F Codd was a Computer Scientist who invented Relational model for Database management. Based on relational model, Relation database was created. Codd proposed 13 rules popularly known as Codd's 12 rules to test DBMS's concept against his relational model. Codd's rule actually define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS). Till now, there is hardly any commercial product that follows all the 13 Codd's rules. Even Oracle follows only eight and half out(8.5) of 13. The Codd's 12 rules are as follows.

Rule zero

This rule states that for a system to qualify as an RDBMS, it must be able to manage database entirely through the relational capabilities.

Rule 1 : Information rule

All information(including meta-data) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.

Rule 2 : Guaranteed Access

Each unique piece of data(atomic value) should be accessible by :
Table Name + primary key(Row) + Attribute(column).

NOTE : Ability to directly access via POINTER is a violation of this rule.

Rule 3 : Systemetic treatment of NULL

Rule 3 : Systemetic treatment of NULL

Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Primary key must not be null. Expression on NULL must give null.

Rule 4 : Active Online Catalog

Database dictionary(catalog) must have description of Database. Catalog to be governed by same rule as rest of the database. The same query language to be used on catalog as on application database.

Rule 5 : Powerful language

One well defined language must be there to provide all manners of access to data. Example: SQL. If a file supporting table can be accessed by any manner except SQL interface, then its a violation to this rule.

Rule 6 : View Updation rule

All view that are theoretically updatable should be updatable by the system.

Rule 7 : Relational Level Operation

There must be Insert, Delete, Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

Rule 8 : Physical Data Independence

The physical storage of data should not matter to the system. If say, some file supporting table were renamed or moved from one disk to another, it should not effect the application.

Rule 9 : Logical Data Independence

If there is change in the logical structure(table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

Rule 10 : Integrity Independence

The database should be able to con-force its own integrity rather than using other programs. Key and Check constraints, trigger etc should be stored in Data Dictionary. This also make *RDBMS* independent of front-end.

Rule 11 : Distribution Independence

A database should work properly regardless of its distribution across a network. This lays foundation of distributed database.

Rule 12 : Non-subversion rule

If low level access is allowed to a system it should not be able to subvert or bypass integrity rule to change data. This can be achieved by some sort of looking or encryption.

13.13 Logical database concepts : Normalization, Entities, attributes, relations

13.13.1 Normalization Rule: Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

There are a few rules for database normalization. Each rule is called a “normal form.” If the first rule is observed, the database is said to be in “first normal form.” If the first three rules are observed, the database is considered to be in “third normal form.” Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

Normalization rule are divided into following normal form.

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF

First Normal Form (1NF)

A row of data cannot contain repeating group of data i.e each column must have a unique value. Each row of data must have a unique identifier i.e *Primary key*. For example consider a table which is not in First normal form

Student Table :

| S_id | S_Name | S_Subject |
|------|------------|------------------|
| 401 | Daryl | Maths |
| 401 | Daryl | Maths |
| 402 | Ramesh | Physics |
| 403 | Rakshana | Maths |
| 404 | Nakshatria | Computer Science |

You can clearly see here that student name Daryl is used twice in the table and subject *maths* is also repeated. This violates the *First Normal form*. To reduce above table to *First Normal form* break the table into two different tables

New Student Table :

| S_id | S_Name | S_Subject |
|------|------------|------------------|
| 401 | Daryl | Maths |
| 402 | Ramesh | Physics |
| 403 | Rakshana | Maths |
| 404 | Nakshatria | Computer Science |

Subject Table :

Fig 13.18 1NF Student table with S-id,S_name,S_subject

| Subject_id | S_id | S_Subject |
|------------|------|------------------|
| 35 | 401 | Maths |
| 35 | 401 | Maths |
| 33 | 403 | Physics |
| 34 | 404 | Chemistry |
| 41 | 405 | Computer Science |

In Student table concatenation of subject_id and student_id is the Primary key. Now both the Student table and Subject table are normalized to first normal form

Second Normal Form (2NF)

A table to be normalized to Second Normal Form should meet all the needs of First Normal Form and there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails Second normal form. For example, consider a table which is not in Second normal form.

Library Table : Fig 13.19 Student table with 1NF rule rewritten

| Library_id | S_Name | Issue_id | Issue_name | Book_detail |
|------------|--------|----------|------------|-------------|
| 101 | RAMU | 10 | Rakesh | C++ |
| 102 | RAMU | 11 | Rakesh | Java |
| 103 | Zama | 12 | Gopal | MATHS |
| 104 | SATISH | 13 | Gopal | MATHS |

In Library table concatenation of Library_id and issue_id is the primary key. This table is in First Normal form but not in Second Normal form because there are partial dependencies of columns on primary key. S_Name is only dependent on Library_id, Issue_name is dependent on Issue_id and there is no link between Book_detail and S_name.

To reduce Library table to Second Normal form break the table into following three different tables.

| Library_id | S_Name |
|------------|--------|
| 101 | RAMU |
| 102 | RAMU |
| 103 | Zama |
| 104 | SATISH |

Issue_Detail Table :

| Issue_id | Issue_name |
|----------|------------|
| 10 | Rakesh |
| 11 | Rakesh |
| 12 | Gopal |
| 13 | Gopal |

Book_Detail Table :

| Library_id | Issue_id | Book_detail |
|------------|----------|-------------|
| 101 | 10 | C++ |
| 102 | 11 | Java |
| 103 | 12 | MATHS |
| 104 | 13 | MATHS |

Now all these three table comply with Second Normal form.

| Library_id S_Name | | Issue_Detail Table : | |
|---------------------|----------|----------------------|------------|
| Library_id | S_Name | Issue_id | Issue_name |
| 101 | RAMU | 10 | Rakesh |
| 102 | RAMU | 11 | Rakesh |
| 103 | Zama | 12 | Gopal |
| 104 | SATISH | 13 | Gopal |
| Book_Detail Table : | | | |
| Library_id | Issue_id | Book_detail | |
| 101 | 10 | C++ | |
| 102 | 11 | Java | |
| 103 | 12 | MATHS | |
| 104 | 13 | MATHS | |

Fig 13.20 2NF Library Table with primary key and Issue key

Now all these three table comply with Second Normal form.

Third Normal Form (3NF)

Third Normal form applies that every non-prime attribute of table must be dependent on primary key. The transitive functional dependency should be removed from the table. The table must be in Second Normal form.

For example, consider a table with following fields.

Student_Detail Table :

| | | | | | | |
|------------|--------------|-----|--------|------|-------|-----|
| Student_id | Student_name | DOB | Street | city | State | Pin |
|------------|--------------|-----|--------|------|-------|-----|

In this table Student_id is Primary key, but street, city and state depends upon pin. The dependency between pin and other fields is called transitive dependency. Hence to apply 3NF, we need to move the street, city and state to new table, with pin as primary key.

New Student_Detail Table :

| | | | |
|------------|--------------|-----|-----|
| Student_id | Student_name | DOB | pin |
|------------|--------------|-----|-----|

Address Table :

| | | | |
|-----|--------|------|-------|
| pin | Street | city | state |
|-----|--------|------|-------|

The advantage of removing transitive dependency is,

- * Amount of data duplication is reduced.
- * Data integrity achieved.

Boyce and Codd Normal Form (BCNF)

BCNF is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

- When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF.
- 3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys
- i.e. composite candidate keys with at least one attribute in common.

- BCNF is based on the concept of a *determinant*.
- A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent.
- A relation is in BCNF is, and only if, every determinant is a candidate key.

Consider the following relation and determinants.

$R(\underline{a}, \underline{b}, c, d)$

$a, c \rightarrow b, d$

$a, d \rightarrow b$

Here, the first determinant suggests that the primary key of R could be changed from a,b to a,c. If this change was done all of the non-key attributes present in R could still be determined, and therefore this change is legal. However, the second determinant indicates that a,d determines b, but a,d could not be the key of R as a,d does not determine all of the non key attributes of R (it does not determine c). We would say that the first determinate is a candidate key, but the second determinant is not a candidate key, and thus this relation is not in BCNF (but is in 3rd normal form).

„A relation $R(X)$ is in Boyce–Codd Normal Form if for every non-trivial functional dependency $Y \rightarrow Z$ defined on it, Y contains a key K of $R(X)$. That is, Y is a superkey for $R(X)$.

13.13.2 Entity-Relationship Diagram

ER-Diagram is a visual representation of data that describes how data is related to each other.

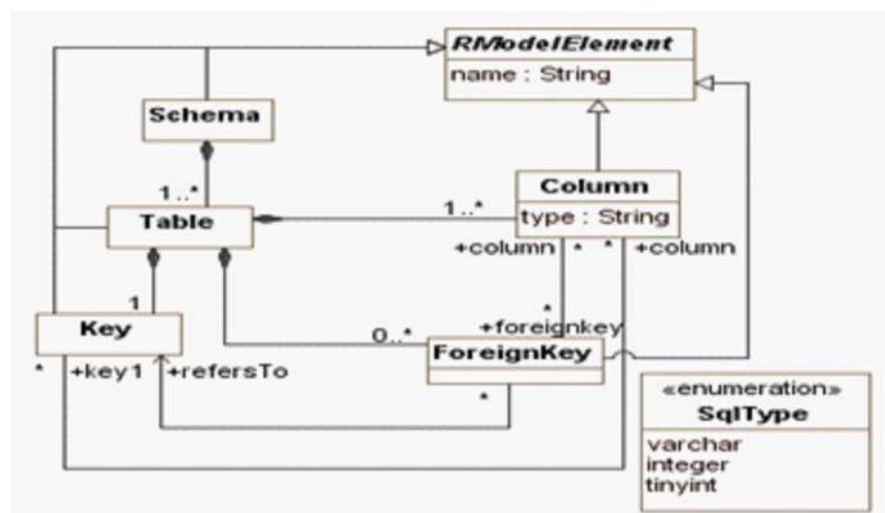


Fig 13.21 Entity-Relationship Symbols

Components of E-R Diagram

The E-R diagram has three main components.

1) Entity

An Entity can be any object, place, person or class. In E-R Diagram, an entity is represented using rectangles. Consider an example of an Organization. Employee, Manager, Department, Product and many more can

be taken as entities from an Organization.

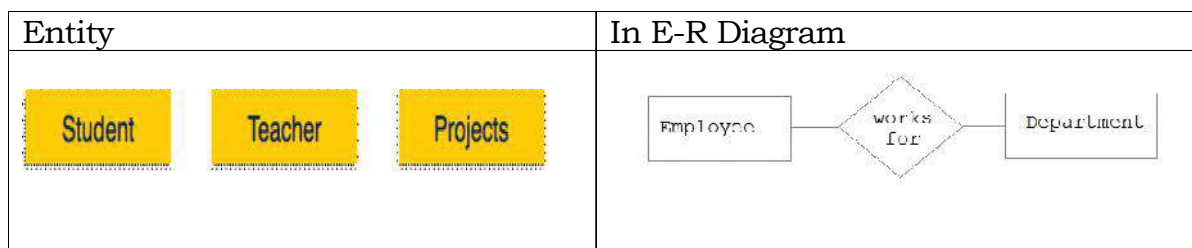
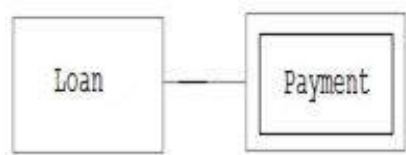


Fig 13.22 Entity and example



Weak entity is an entity that depends on another entity. Weak entity doesn't have key attribute of their own. Double rectangle represents weak entity.

Fig 13.23 Entity with loan and payment

2) Attribute

An Attribute describes a property or characteristic of an entity. For example, Name, Age, Address etc. can be attributes of a Student. An attribute is represented using eclipse.

Attribute

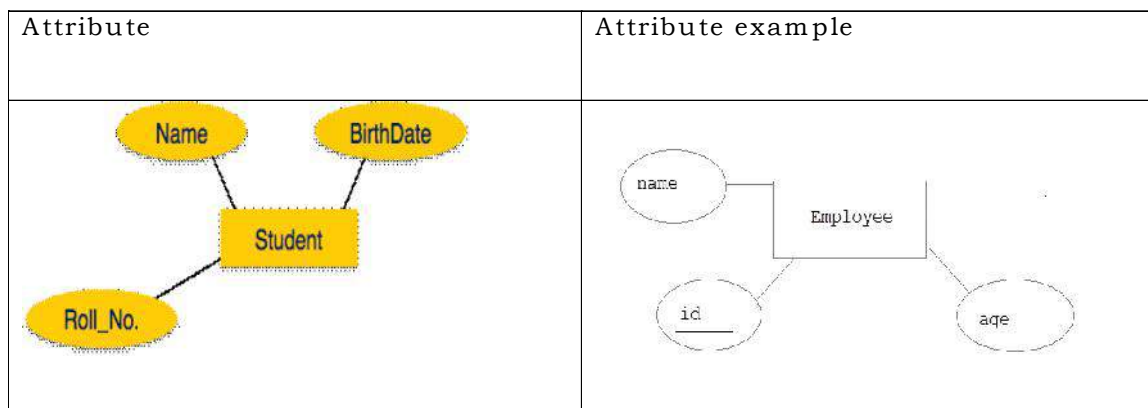
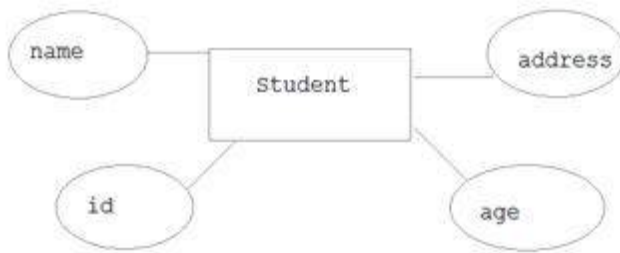
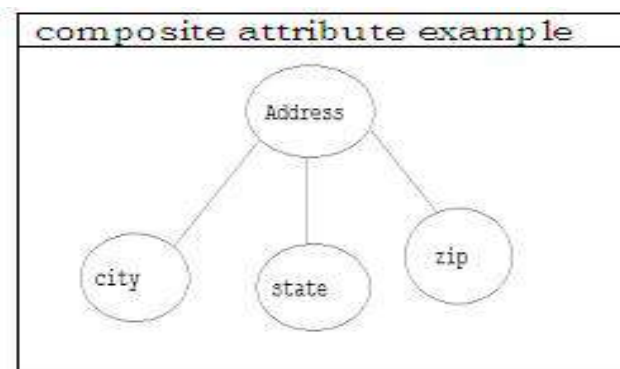


Fig 13.24 Attribute with example



Key Attribute Key attribute represents the main characteristic of an Entity. It is used to represent Primary key. Ellipse with underlying lines represent Key Attribute.

Fig 13.25 Attribute key with primary key



Composite Attribute : An attribute can also have their own attributes. These attributes are known as Composite attribute.

composite attribute example

Fig 13.26 Composite attribute

3) Relationship

A Relationship describes relations between entities. Relationship is represented using diamonds.

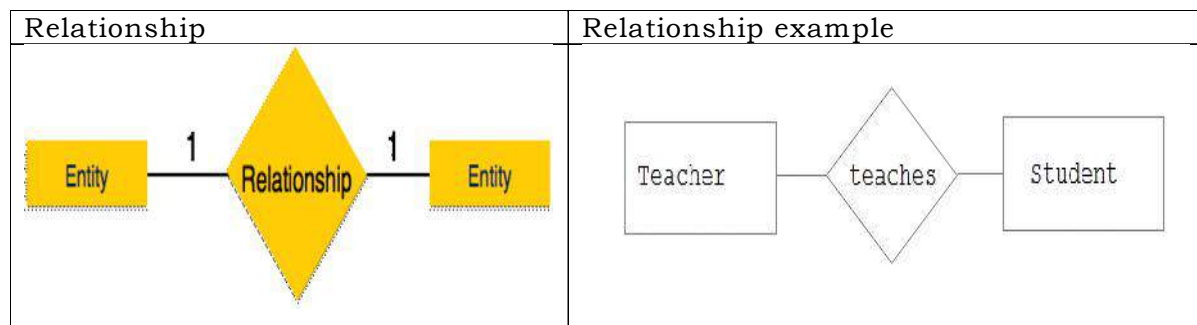


Fig 13.27 Relationship with example

There are three types of relationship that exist between Entities.

- * Binary Relationship
- * Recursive Relationship
- * Ternary Relationship

Binary Relationship

Binary Relationship means relation between two Entities. This is further divided into three types.

1. **One to One** : This type of relationship is rarely seen in real world.

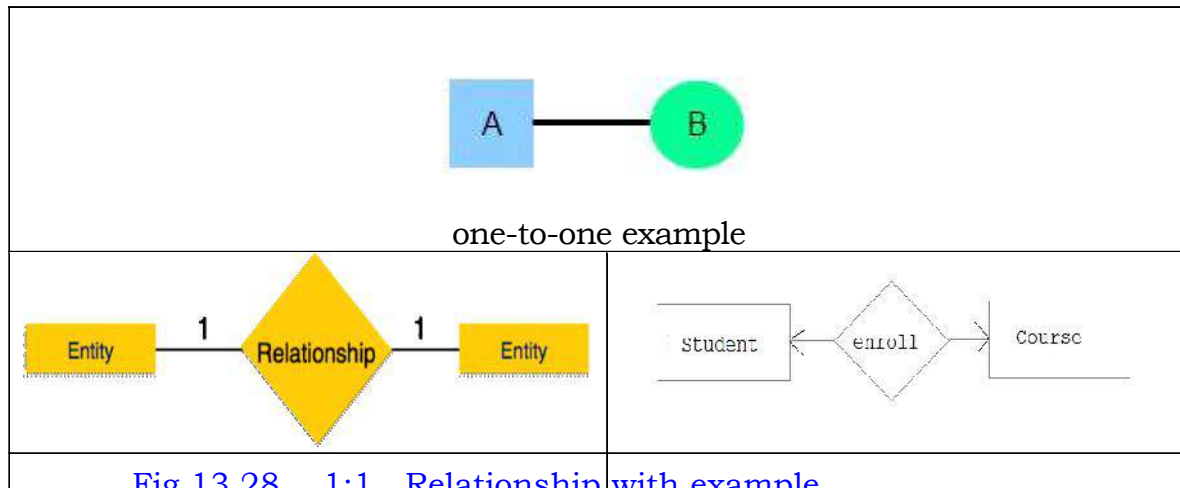


Fig 13.28 1:1 Relationship with example

The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in relationship.

2. **One to Many** : It reflects business rule that one entity is associated with many number of same entity. For example, Student enrolls for only one Course but a Course can have many Students.

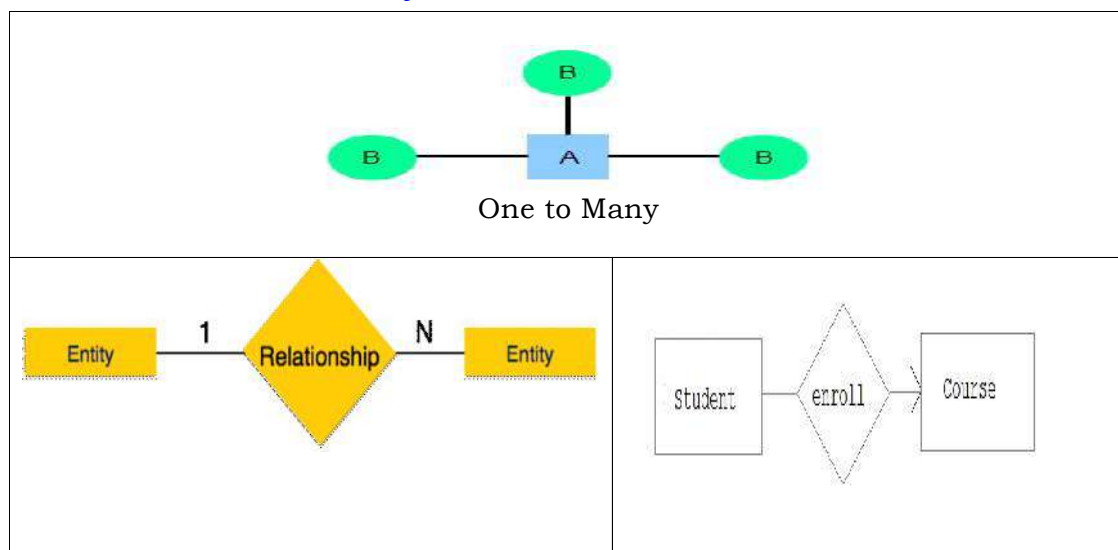


Fig 13.29 1:M Relationship with example

The arrows in the diagram describes that one student can enroll for only one course.

3. **Many to Many :** The above diagram represents that many students can enroll for more than one courses.

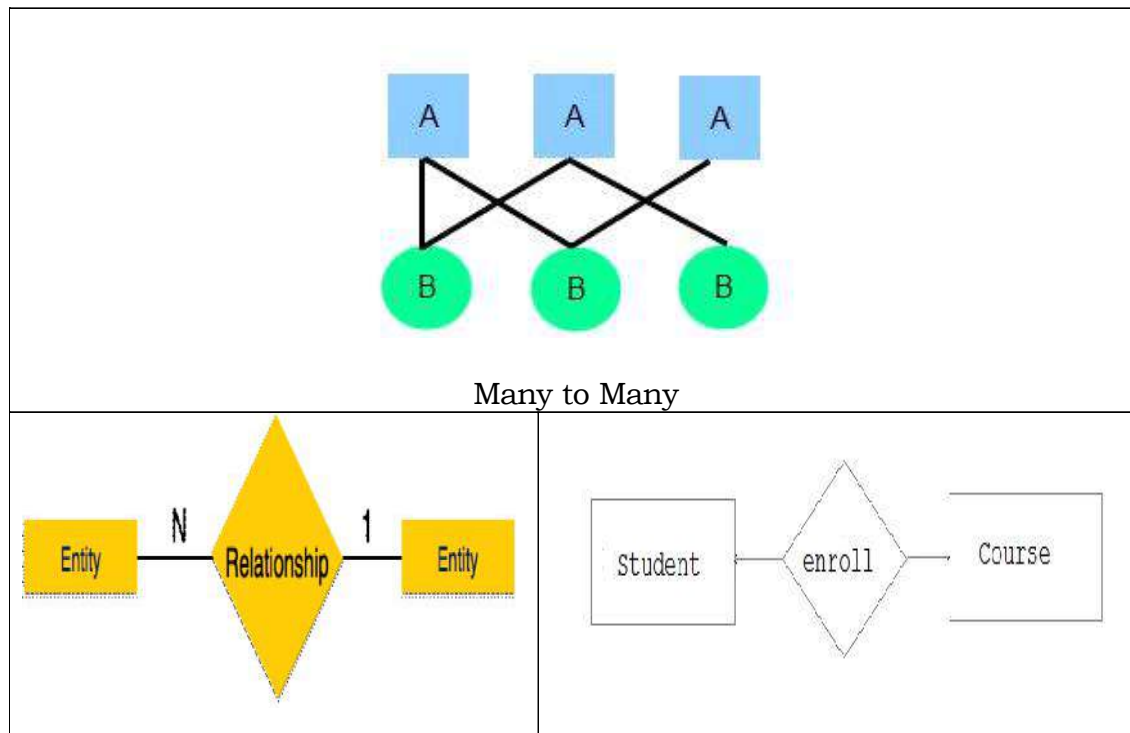


Fig 13.30 M:M Relationship with example

PARTICIPATION CONSTRAINTS

- **Total Participation:** Each entity in the entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation:** Not all entities are involved in the relation ship. Partial participation is represented by single line.

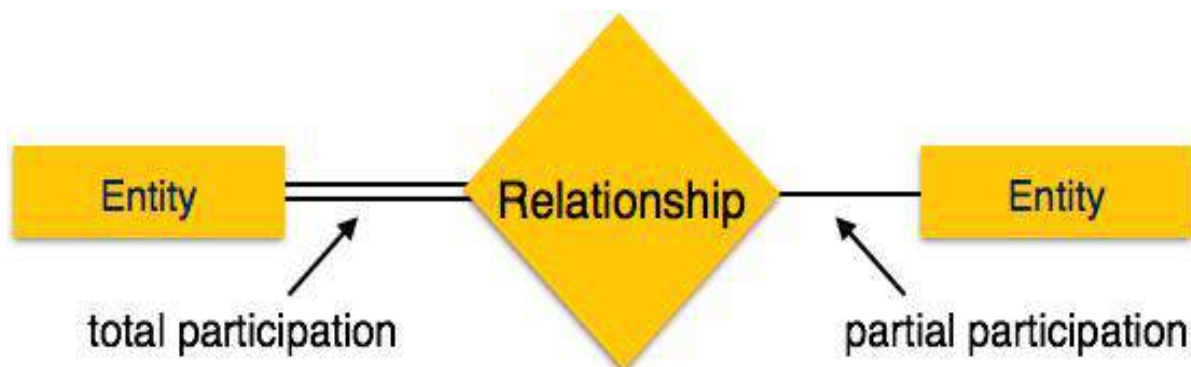
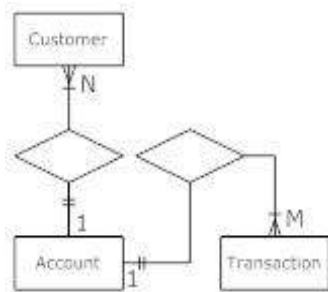


Fig 13.31 Total participation and partial participation

13.13.3 Cardinality



Cardinality specifies how many instances of an entity relate to one instance of another entity.

Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.

Fig 13.32 Cardinality

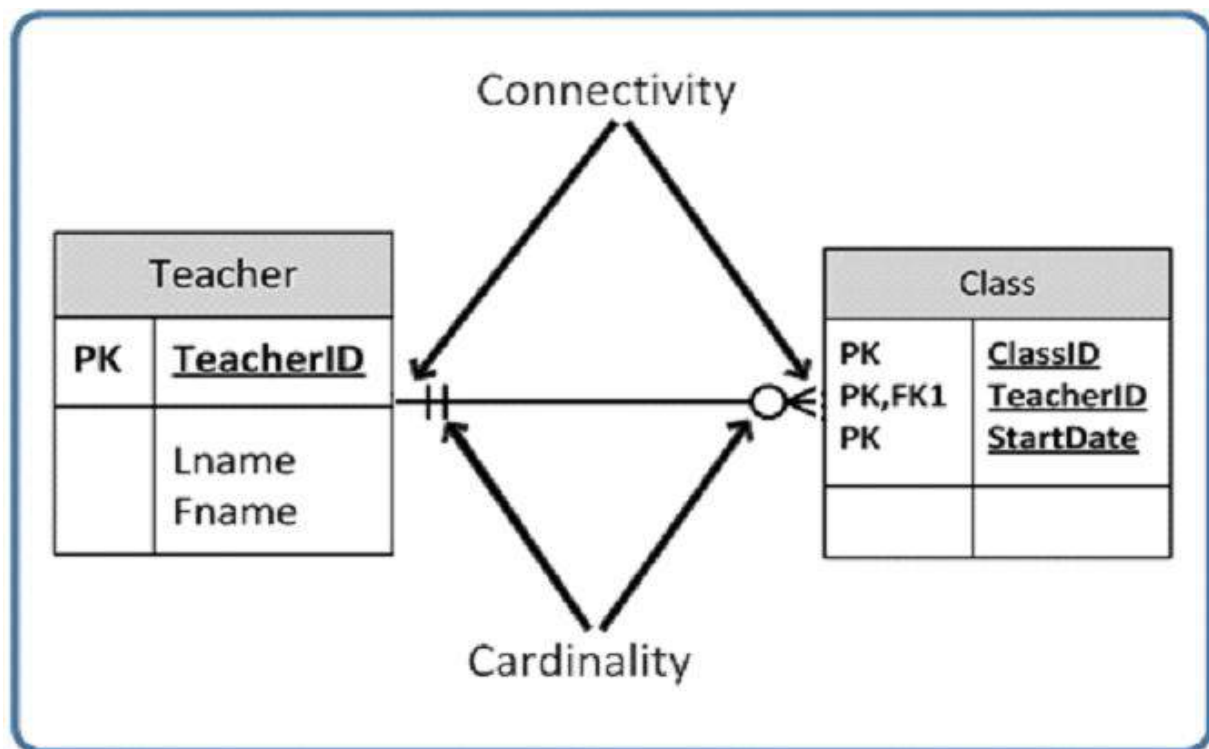


Fig 13.33 Cardinality Notation with examples

Cardinality Notation Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships. When the minimum number is zero, the relationship is usually called optional and when the minimum number is one or more, the relationship is usually called mandatory.

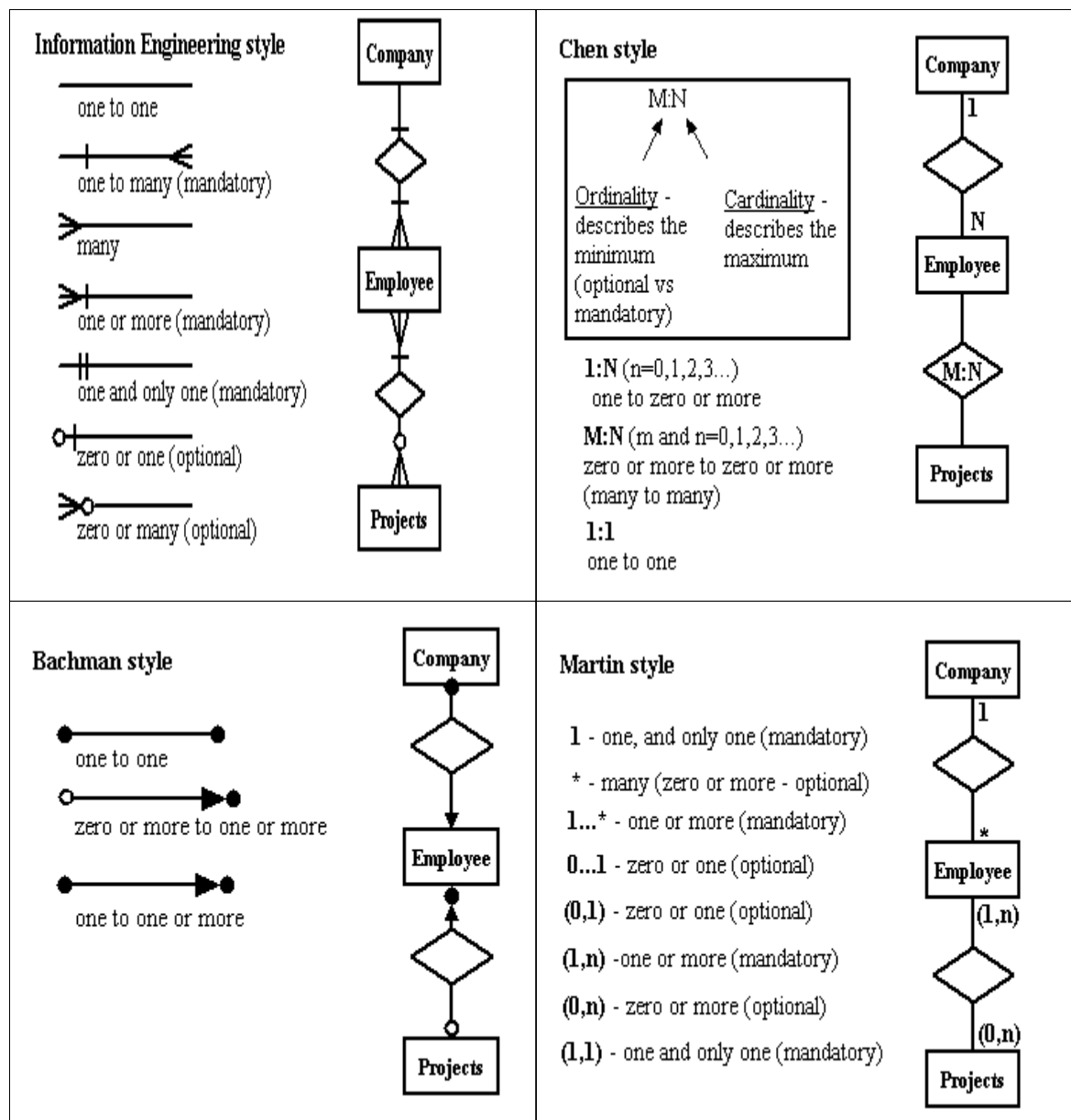


Fig 13.34 Cardinality Notation

Recursive Relationship

Fig 13.35 Recursive Relationship

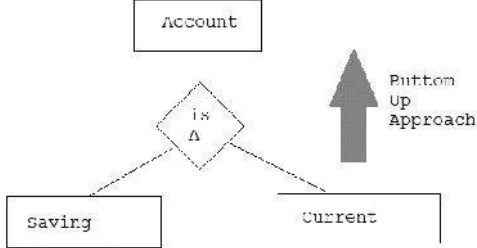
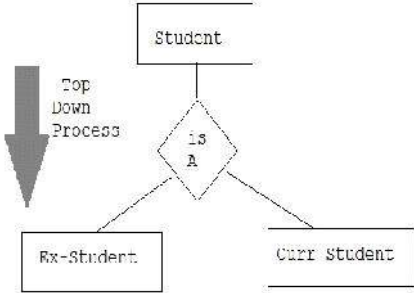
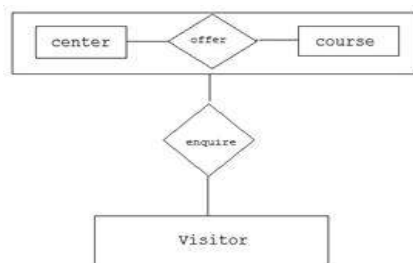
| Generalization | Specialization |
|--|---|
| Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entity to make further higher level entity. | *Specialization* is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, some higher level entities may not have lower-level entity sets at all. |
|  |  |

Fig13.36 Generalization & Specialization



When an Entity is related with itself it is known as Recursive Relationship.

Fig13.37 Aggregation

Aggregation : Aggregation is a process when relation between two entity is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.

13.14 Keys

The word “key” is used in the context of relational database design. They are used to establish and identify relation between tables. The key is a set of one or more columns whose combined values are unique among all occurrences in a given table.

| Table | Employees | Relation | | |
|-------------------------|--------------------------------------|----------|--------|--------|
| Employee_id | Name | Age | city | Salary |
| 1 | Rajappa | 42 | Tumkur | 42000 |
| 2 | Naveen | 35 | Bidar | 35000 |
| 2 | Ravindra | 42 | Sagar | 45000 |
| Domain | | | | |
| Relation/table | Employees | | | |
| Attribute | Employee_id, Name, age, city, salary | | | |
| tuple | 2, naveen, 35, Bidar, 35000 | | | |
| domain | 42000, 35000, 45000 | | | |
| candidate Keys | Employee_id, Name | | | |
| Primary Key | Employee_id | | | |
| Secondary/Alternate key | Name | | | |

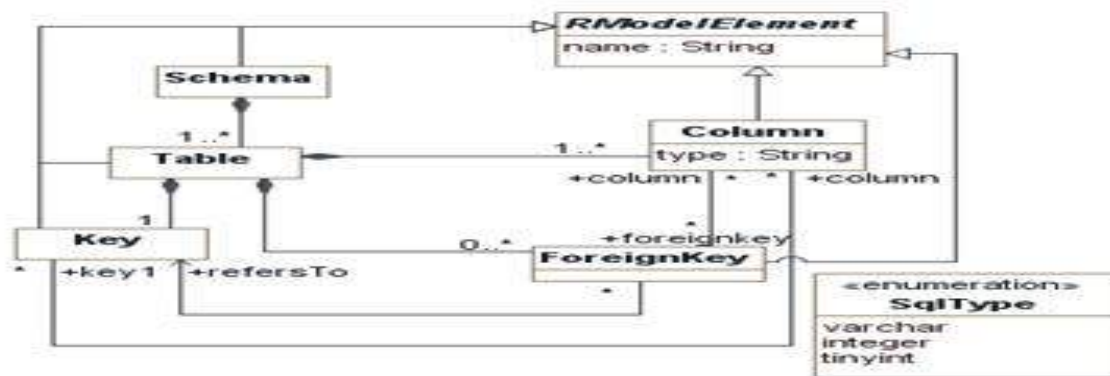


Fig 13.38 Database terms with example

There are various types of relational keys:

1. **Candidate key(ck):** A Candidate key is any set of one or more columns whose combined values are unique among all occurrences (ie tuples or rows). Since a null value is not guaranteed to be unique, no component of a candidate key is allowed to be null.
2. **Primary key(pk):** Primary key is a candidate key that is most appropriate to become the main key of the table. Primary key is a key that uniquely identifies each record in a table. Example student_id, Employee_id, Bank_account_no, Transfer_certificate_id, Driving_licence_no etc., by which only one value can exist, no duplication can exist.

Primary Key

| R_ID | S_NAME | AGE | COURSE | ADDRESS |
|------|--------|-----|--------|---------|
| | | | | |

3. **Alternate key/Secondary key(sk)**: The alternate keys of any table are simply those candidate keys which are not currently selected as the primary key. An alternative key is a function of all candidate keys minus the primary key.
5. **Super Key** : A superkey is basically all sets of columns for which no two rows share the same values for those sets. An attribute or set of attributes that uniquely identifies a tuple within a relation/table. Super Key is a superset of Candidate key.
6. **Foreign key(fk)** :A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.

| Table | Employees | | | | |
|-------------|-----------|-----|--------|--------|-------------|
| Employee_id | Name | Age | city | Salary | Car_loan_id |
| 1 | Rajappa | 42 | Tumkur | 42000 | 585 |

| Table | BMWcars | | | | |
|-------------|---------|-------------|------|-----------|---------|
| Car_loan_id | Model | Loan amount | EMI | No of EMI | Balance |
| 585 | Basic | 1800000 | 8000 | 225 | 1000000 |

Fig 13.39 Foreign key with example

1. **Composite Key** : Key that consists of two or more attributes that uniquely identify an entity occurrence is called Composite key. But any attribute that makes up the Composite key is not a simple key in its own. Example: Consider a Relation or Table R1. Let A,B,C,D,E are the attributes of this relation.

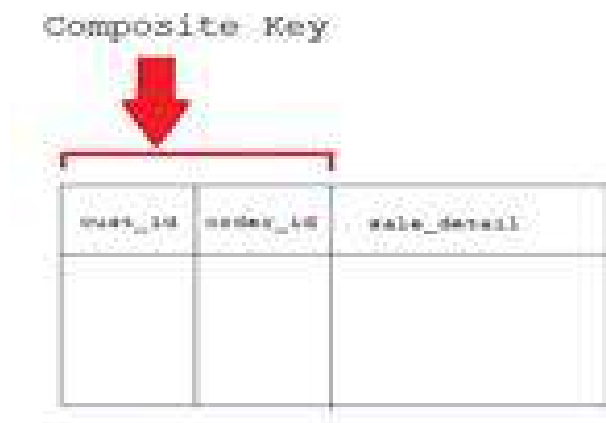


Fig.14.47 Composite Key

R(A,B,C,D,E)

A?BCDE This means the attribute 'A' uniquely determines the other attributes B,C,D,E.

BC?ADE This means the attributes 'BC' jointly determines all the other attributes A,D,E in the relation.

Primary Key :A

Candidate Keys :A, BC

Super Keys : A,BC,ABC,AD

Note : ABC,AD are not Candidate Keys since both are not minimal super keys.

13.15 Relational Algebra

In order to implement a DBMS, there must exist a set of rules which state how the database system will behave. For instance, somewhere in the DBMS must be a set of statements which indicate that when someone inserts data into a row of a relation, it has the effect which the user expects. One way to specify this is to use words to write an 'essay' as to how the DBMS will operate, but words tend to be imprecise and open to interpretation. Instead, relational databases are more usually defined using Relational Algebra.

Relational Algebra is :

- the formal description of how a relational database operates
- an interface to the data stored in the database itself
- the mathematics which underpin SQL operations

Operators in relational algebra are not necessarily the same as SQL operators, even if they have the same name. For example, the SELECT statement exists in SQL, and also exists in relational algebra. These two uses of SELECT are not the same. The DBMS must take whatever SQL statements the user types in and translate them into relational algebra operations before applying them to the database.

Terminology

- Relation – a set of tuples.
- Tuple – a collection of attributes which describe some real world entity.
- Attribute – a real world role played by a named domain.
- Domain – a set of atomic values.
- Set – a mathematical definition for a collection of objects which contains no duplicates.

Operators – Write

- **INSERT** – provides a list of attribute values for a new tuple in a relation. This operator is the same as SQL.
- **DELETE** – provides a condition on the attributes of a relation to determine which tuple(s) to remove from the relation. This operator is the same as SQL.
- **MODIFY** – changes the values of one or more attributes in one or more tuples of a relation, as identified by a condition operating on the attributes of the relation. This is equivalent to SQL UPDATE.

Operators – Retrieval

There are two groups of operations:

- Mathematical set theory based relations:
UNION, INTERSECTION, DIFFERENCE, and CARTESIAN PRODUCT.

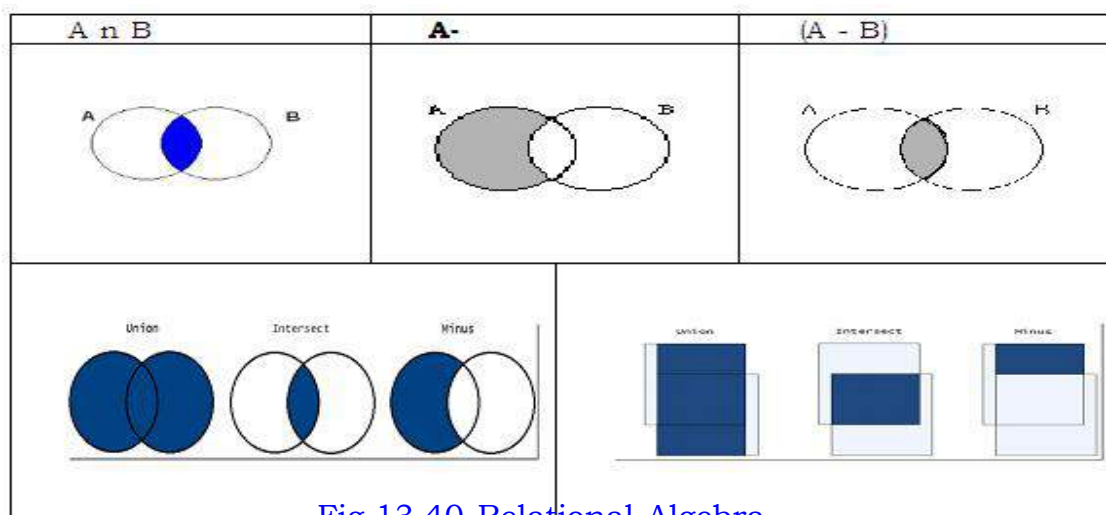


Fig.13.40 Relational Algebra

- Special database operations:
SELECT (not the same as SQL SELECT), PROJECT, and JOIN.

Relational SELECT

SELECT is used to obtain a subset of the tuples of a relation that satisfy a *select condition*.

For example, find all employees born after 14th Feb 2014:

```
SELECTdob '14/feb/2014'(employee)
```

Relational PROJECT

The PROJECT operation is used to select a subset of the attributes of a relation by specifying the names of the required attributes.

For example, to get a list of all employees surnames and employee numbers:

```
PROJECTsurname, empno(employee)
```

SELECT and PROJECT

SELECT and PROJECT can be combined together. For example, to get a list of employee numbers for employees in department number 1:

PROJECT_{EMPNO} (SELECT_{DEPTNO=1} (EMPLOYEE))

MAPPING THIS BACK TO SQL GIVEN:

SELECT EMPNO
FROM EMPLOYEE
WHERE DEPTNO=1;



Fig.13.41 Select and project
Figure : Mapping select and project

Set Operations – semantics

Consider two relations R and S.

- UNION of R and S

the union of two relations is a relation that includes all the tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

- INTERSECTION of R and S

the intersection of R and S is a relation that includes all tuples that are both in R and S.

- DIFFERENCE of R and S

the difference of R and S is the relation that contains all the tuples that are in R but that are not in S.

SET Operations – requirements

For set operations to function correctly the relations R and S must be union compatible. Two relations are union compatible if

- they have the same number of attributes
- the domain of each attribute in column order is the same in both R and S.

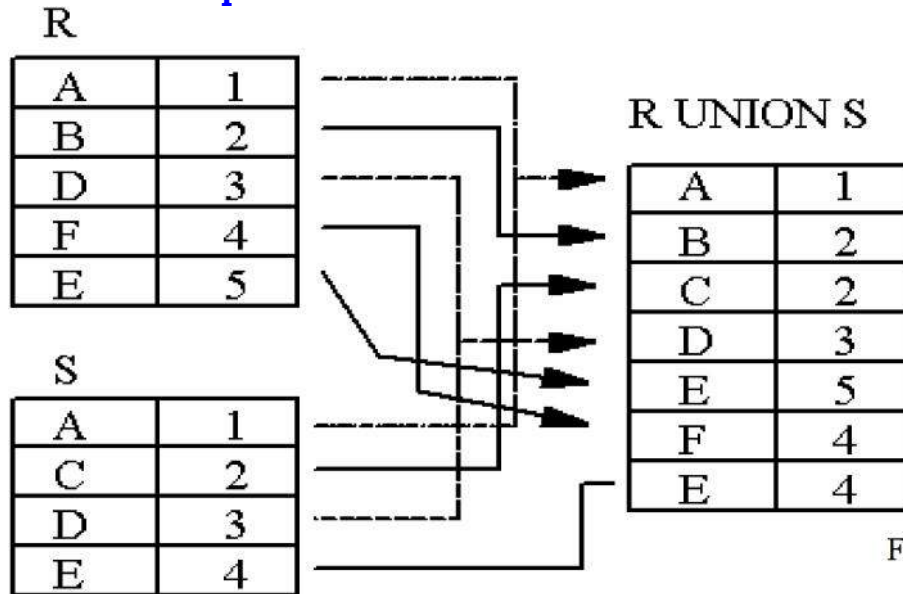
UNION Example

Fig.13.42 Union

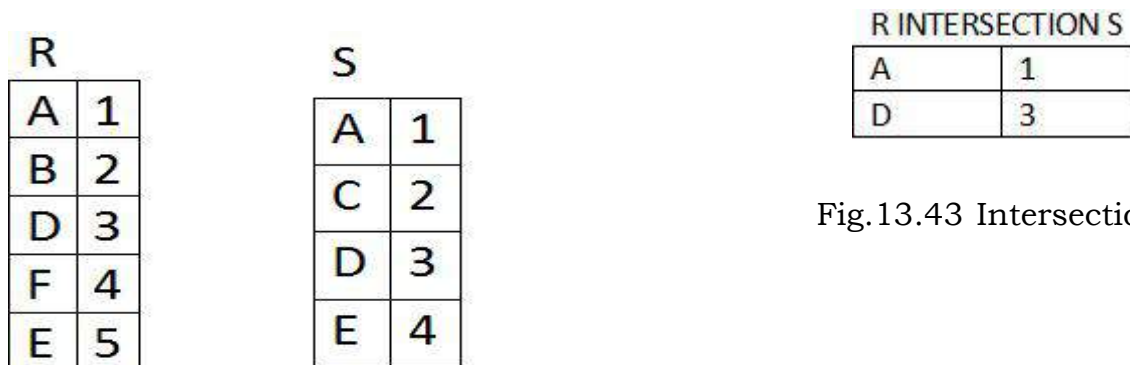
INTERSECTION Example

Fig.13.43 Intersection

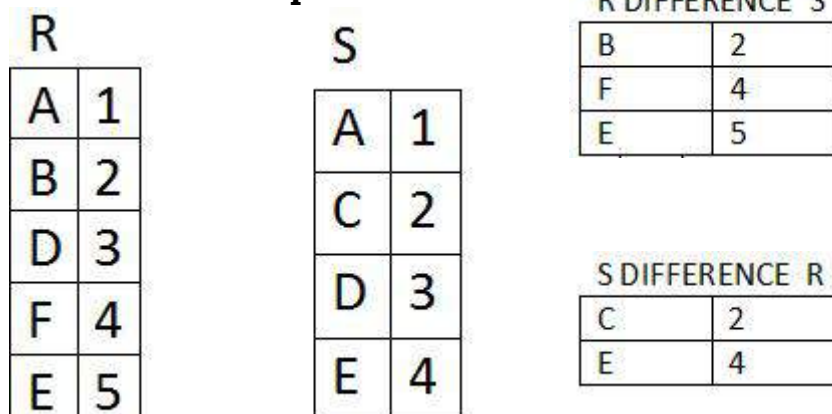
DIFFERENCE Example

Fig.13.44 Difference

CARTESIAN PRODUCT

The Cartesian Product is also an operator which works on two sets. It is sometimes called the CROSS PRODUCT or CROSS JOIN.

It combines the tuples of one relation with all the tuples of the other relation.

CARTESIAN PRODUCT example

| R | | R CROSS S | | | |
|---|---|-----------|---|---|---|
| A | 1 | A | 1 | F | 4 |
| B | 2 | A | 1 | F | 4 |
| D | 3 | A | 1 | F | 4 |
| F | 4 | A | 1 | F | 4 |
| E | 5 | B | 2 | E | 5 |
| S | | B | 2 | A | 1 |
| A | 1 | B | 2 | E | 5 |
| C | 2 | B | 2 | C | 2 |
| D | 3 | B | 2 | D | 3 |
| E | 4 | D | 3 | E | 4 |
| | | D | 3 | A | 1 |
| | | D | 3 | C | 2 |
| | | D | 3 | D | 3 |
| | | D | 3 | E | 4 |

Fig.13.45 Cartesian product

Figure : CARTESIAN PRODUCT

JOIN Operator

JOIN is used to combine related tuples from two relations:

- In its simplest form the JOIN operator is just the cross product of the two relations.
- As the join becomes more complex, tuples are removed within the cross product to make the result of the join more meaningful.
- JOIN allows you to evaluate a join condition between the attributes of the relations on which the join is undertaken.

The notation used is

R JOIN_{join condition} S

| R | ColA | ColB |
|---|------|------|
| A | 1 | |
| B | 2 | |
| D | 3 | |
| F | 4 | |
| E | 5 | |

| S | SColA | SColB |
|---|-------|-------|
| A | 1 | |
| C | 2 | |
| D | 3 | |
| E | 4 | |

JOIN Example

R JOIN_{R.ColA = S.SColA} S

| | | | |
|---|---|---|---|
| A | 1 | A | 1 |
| D | 3 | D | 3 |
| E | 5 | E | 4 |

R JOIN_{R.ColB = S.SColB} S

| | | | |
|---|---|---|---|
| A | 1 | A | 1 |
| B | 2 | C | 2 |
| D | 3 | D | 3 |
| F | 4 | E | 4 |

Figure : JOIN Fig.13.46 Join

Natural Join

Invariably the JOIN involves an equality test, and thus is often described as an equi-join. Such joins result in two attributes in the resulting relation having exactly the same value. A 'natural join' will remove the duplicate attribute(s).

- In most systems a natural join will require that the attributes have the same name to identify the attribute(s) to be used in the join. This may require a renaming mechanism.
- If you do use natural joins make sure that the relations do not have two attributes with the same name by accident.

OUTER JOINS

Notice that much of the data is lost when applying a join to two relations. In some cases this lost data might hold useful information. An outer join retains the information that would have been lost from the tables, replacing missing data with nulls.

There are three forms of the outer join, depending on which data is to be kept.

- LEFT OUTER JOIN – keep data from the left-hand table
- RIGHT OUTER JOIN – keep data from the right-hand table
- FULL OUTER JOIN – keep data from both tables

OUTER JOIN example 1

| R | | LEFT OUTER JOIN | | R.ColA = S.SColA | | S _R | ColA |
|-------|--|------------------|---|------------------|---|----------------|-------|
| ColB | | A | 1 | A | 1 | | A |
| 1 | | D | 3 | D | 3 | | B |
| 2 | | E | 5 | E | 4 | | D |
| 3 | | B | 2 | - | - | | F |
| 4 | | F | 4 | - | - | | E |
| 5 | | | | | | | |
| S | | RIGHT OUTER JOIN | | R.ColA = S.SColA | | S _S | SColA |
| SColB | | A | 1 | A | 1 | | A |
| 1 | | D | 3 | D | 3 | | C |
| 2 | | E | 5 | E | 4 | | D |
| 3 | | - | - | C | 2 | | E |
| 4 | | | | | | | |

Fig.13.47 Outer Join left/right *OUTER JOIN (left/right)*

JOIN example 2

| $CoIB$ | R FULL OUTER JOIN $R.CoIA = S.SCoIA$ | | | | S | $CoIA$ |
|--------|--------------------------------------|---|---|---|-----|--------|
| 1 | A | 1 | A | 1 | | A |
| 2 | D | 3 | D | 3 | | B |
| 3 | E | 5 | E | 4 | | D |
| 4 | B | 2 | - | - | | F |
| 5 | F | 4 | - | - | | E |
| | - | - | C | 2 | | |

| $SCoIB$ | S | $SCoIA$ |
|---------|-----|---------|
| 1 | | A |
| 2 | | C |
| 3 | | D |
| 4 | | E |

Fig.13.48 Outer Join Full *OUTER JOIN (full)*

Consider the following SQL to find which departments have had employees on the 'Further Accounting' course.

```
SELECT DISTINCT dname
FROM department, course, empcourse, employee
WHERE cname = 'Further Accounting'
AND course.courseno = empcourse.courseno
AND empcourse.empno = employee.empno
AND employee.depno = department.depno;
```

The equivalent relational algebra is

```
PROJECTdname (department JOINdepno = depno (
PROJECTdepno (employee JOINempno = empno (
PROJECTempno (empcourse JOINcourseno = courseno (
PROJECTcourseno (SELECTcname = 'Further Accounting' course) )) )) )
```

Symbolic Notation

From the example, one can see that for complicated cases a large amount of the answer is formed from operator names, such as PROJECT and JOIN. It is therefore commonplace to use symbolic notation to represent the operators.

- SELECT $\rightarrow \sigma$ (sigma)
- PROJECT $\rightarrow \pi$ (pi)
- PRODUCT $\rightarrow \times$ (times)
- JOIN $\rightarrow \bowtie$ (bow-tie)

- UNION \rightarrow \cup (cup)
- INTERSECTION \rightarrow \cap (cap)
- DIFFERENCE \rightarrow $-$ (minus)
- RENAME \rightarrow ρ (rho)

Usage

The symbolic operators are used as with the verbal ones. So, to find all employees in department 1:

SELECT_{depno = 1}(employee)
 becomes $\sigma_{\text{depno} = 1}$ (employee)

Conditions can be combined together using \wedge (AND) and \vee (OR). For example, all employees in department 1 called 'URS':

SELECT_{depno = 1 \wedge surname = 'URS'}(employee)
 becomes $\sigma_{\text{depno} = 1 \wedge \text{surname} = \text{'URS'}}$ (employee)

The use of the symbolic notation can lend itself to brevity. Even better, when the JOIN is a natural join, the JOIN condition may be omitted from $|x|$. The earlier example resulted in:

PROJECT_{dname} (department JOIN_{depno = depno} (
 PROJECT_{depno} (employee JOIN_{empno = empno} (
 PROJECT_{empno} (empcourse JOIN_{courseno = courseno} (
 PROJECT_{courseno} (SELECT_{cname = 'Further Accounting'}
 course))))))

becomes

σ_{dname} (department $| \times |$ (
 σ_{depno} (employee $| \times |$ (
 σ_{empno} (empcourse $| \times |$ (
 σ_{courseno} ($\sigma_{\text{cname} = \text{'Further Accounting'}}$ course)))))

Rename Operator

The rename operator returns an existing relation under a new name.

$\tilde{r}_A(B)$ is the relation B with its name changed to A. For example, find the employees in the same Department as employee 3.

$$\tilde{r}_{\text{emp2.surname, emp2.forenames}} \left(\sigma_{\text{employee.empno} = 3 \wedge \text{employee.depno} = \text{emp2.depno}} \left(\text{employee} \times (\tilde{r}_{\text{emp2}} \text{employee}) \right) \right)$$

Derivable Operators

- Fundamental operators: σ , θ , \times , $*$, $-$, \tilde{r}
- Derivable operators: $| \times |$, $| \cup |$

Equivalence

$$A | \times |_c B \hat{=} \theta_{a_1, a_2, \dots, a_N} (\sigma_c (A \times B))$$

- where c is the join condition (eg $A.a_1 = B.a_1$),
- and a_1, a_2, \dots, a_N are all the attributes of A and B without repetition.

C is called the join-condition, and is usually the comparison of primary and foreign key. Where there are N tables, there are usually N-1 join-conditions. In the case of a natural join, the conditions can be missed out, but otherwise missing out conditions results in a ptimizat product (a common mistake to make).

Equivalences

The same relational algebraic expression can be written in many different ways. The order in which tuples appear in relations is never significant.

- $A \times B \hat{=} B \times A$
- $A \cup B \hat{=} B \cup A$
- $A * B \hat{=} B * A$
- $(A - B)$ is not the same as $(B - A)$

- $\sigma_{c1}(\sigma_{c2}(A)) \hat{=} \sigma_{c2}(\sigma_{c1}(A)) \hat{=} \sigma_{c1 \wedge c2}(A)$
- $\sigma_{a1}(A) \hat{=} \sigma_{a1}(\sigma_{a1,etc}(A))$

where etc represents any other attributes of A.

- many other equivalences exist.

While equivalent expressions always give the same result, some may be much easier to evaluate than others.

When any query is submitted to the DBMS, its query optimizer tries to find the most efficient equivalent expression before evaluating it.

| Comparing RA and SQL | |
|--|---|
| Relational algebra: | SQL: |
| <ul style="list-style-type: none"> • Is closed (the result of every expression is a relation) • Has a rigorous foundation • Has simple semantics • Is used for reasoning, query optimization, etc. | <ul style="list-style-type: none"> • Is a superset of relational algebra • Has convenient formatting features, etc. • Provides aggregate functions • Has complicated semantics • Is an end-user language |
| <p>Any relational language as powerful as relational algebra is called relationally complete.</p> <p>A relationally complete language can perform all basic, meaningful operations on relations.</p> | <p>SQL is a superset of relational algebra, it is also relationally complete.</p> |

13.16 Data warehouse

A data warehouse is a repository of an organization's electronically stored data. Data warehouses are designed to facilitate reporting and supporting data analysis. The concept of data warehouses was introduced in the late 1980's. The concept was introduced to meet the growing demands for management information and analysis that could not be met by operational systems.

Separate computer databases began to be built that were specifically designed to support management information and analysis purposes. These data warehouses

were able to bring in data from a range of different data sources, such as, mainframe computer, minicomputer, as well as personal computer and office automation software such as spreadsheets and integrate this information in a single place. This capability, coupled with user-friendly reporting tools, and freedom from operational impacts has led to a growth of this type of computer system.

Data ware house have evolved though several fundamental stages like:

Offline operational databases – Data warehouse in this initial stage are developed by simply copying the database of an operational system to an off-line server where the processing load of reporting does not impact on the operational system's performance.

Offline data warehouse – Database warehouses in this stages of evolution are updated on regular time cycle(usually daily, weekly or monthly) form operational systems and the data is stored in a integrated reporting-oriented data structure.

Real Time data warehouse – Data warehouses are updated on transaction or event basis, event time an operational system performs a transaction.

Integrated data warehouses – Data warehouses used to generate activity or transactions that are passed back into the operational systems for use in the daily activity of the organization.

Components of data warehouses

Data Sources: Data sources refer to any electronic repository of information that contains data of interest for management use or analytics. From mainframe(IBM DB2,ISAM,Adabas, etc.), client-server databases (e.g Oracle database, Informix, Microsoft SQL Server etc.), PC databases (e.g Microsoft Access), and ESS and other electronic store of data. Data needs to be passed from these to systems to the data warehouse either on the transaction-by-transaction basis for real-time data warehouses or on a regular cycle(e.g daily or weekly) for offline data warehouses.

Data transformation: The data transformation layer receives data from the data sources, cleaned and standardizes and loads it into the data repository. This is often called “staging” data as data often passes through a temporary database whilst it is being transformed. This activity of transformation data can be performed either by manually created code or a specific type of software could be used called an Extract, Transform and load(ETL) tool.

Reporting : The data in the data warehouses must be available to the organization's staff if the data warehouses is to be useful. There are a very large number of applications that perform this function or reporting can be custom-developed. Some are Business intelligence tools, Executive information systems, Online Analytical processing(OLAP) Tools, Data Mining etc.,

Metadata: Metadata or "Data about data" is used to inform operators and uses of the data warehouses about its status and the information held within the data warehouses.

Operations : Data warehouses operations comprises of the processes of loading, manipulating and extracting data from the data warehouse. Operations also cover users management security, capacity management and related functions.

Optional components: In addition the following components also exist in some data warehouses: 1. Dependent data marts. 2. Logical data marts. 3. Operational data store.

Advantages of data ware houses:

1. Enhance end-user access to reports and analysis of information.
2. Increases data consistency.
3. Increases productivity and decreases computing costs.
4. Able to combine data from different sources, in one place.
5. Data warehouses provide an infrastructure that could support changes to data and replication of the changed data back into the operational systems.

Disadvantages

Extracting, cleaning and loading data could be time consuming.

Data warehouses can get outdated relatively quickly.

Problems with compatibility with systems already in place.

Providing training to end-users.

Security could develop into a serious issue, especially if the data warehouses is internet accessible.

A data warehouses is usually not static and maintenance costs are high.

13.17 Data Mining : Data mining is concerned with the analysis and picking out relevant information. It is the computer, which is responsible for finding the patterns by identifying the underling rules of the features in the data.

Data mining analysis tends to work form the data up and the best techniques are those developed with an orientation towards large volumes of data, making use of as much of the collected data as possible to arrive at reliable conclusions and decisions.

The analysis process starts with a set of data, uses a methodology to develop an optimal representation to the structure of the data during which time knowledge is acquired. Once knowledge has been acquired this can be extended to larger sets of data working on the assumption that the larger data set has a structure similar to the sample data. Again this is analogous to a mining operation where large amounts of low grade materials are sifted through in order to find something of value.

Some of the data mining software's are SPSS, SAS, Think Analytics and G-Sat etc.

The phases start with the raw data and finish with the extracted knowledge which was acquired as a result of the following stages:

Selection- Selecting or segmenting the data according to some criteria e.g. all those people who won a car, in this way subsets of the data can be determined.

Preprocessing – This is the data cleaning stage where certain information is removed which deemed unnecessary and may slow down queries for e.g.: gender of the patient. The data is reconfigured to ensure a consistent format as there is a possibility of inconsistent formats because the data is drawn from several sources e.g. gender may be recorded as F or M also as 1 or 0.

Transformation – The data is not merely transferred, but transformed. E.g.: demographic overlays commonly used in market research. The data is made useable and navigable.

Data mining- This stage is concerned with the extraction of patterns from the data. A pattern can be defined as given a set of facts(data) F , a language L , and some measure of certainty C a pattern is a statement S in L that describes relationships among a subset F_s of F with a certainty c such that S is simpler in some sense than the enumeration of all the facts in F_s .

Interpretation and Evaluation – The patterns identified by the system are interpreted into knowledge which can be used to support human decision-making e.g. prediction and classification tasks, summarizing the content of a database or explaining observed phenomena.

Summary

- > The basic concepts of database that can be used by various users to store and retrieve the data in standardized format.
- > DBMS features, parts, problems and solutions.
- > Three database structures.
- > Entity relations.
- > Various relationships.
- > Keys
- > Database warehouse, Data mining

Review questions**One mark questions**

1. What is data?
2. What is information?
3. What is database?
4. What is a field?
5. What is a record?
6. What is an entity?
7. What is an instance?
8. What is an attribute?
9. What is domain?
10. What is a relation?
11. What is a table?
12. What is normalization?
13. What is a key?
14. Give the symbol notation for project.
15. What is data mining?

Two marks questions

1. How database helps us?
2. How do we get data?
3. Name the data types supported by DBMS.
4. What is generalization?
5. What is specification?
6. What is the difference between serial and direct access file organization?
7. Give the advantages and disadvantages of Index Sequential Access Method.
8. Classify various types of keys used in Database.
9. What is relation Algebra?
10. Give an example for relation **selection** with example.
11. What is Cartesian product?
12. What is Join operation?
13. What is data warehouse?
14. What is data mining?

Three marks questions

1. Mention the applications of database.
2. List different forms of data(any three)
3. Give the difference between Manual and Electronic file systems.
4. Explain Boyce and Codd form (BCNF)
5. Explain any three components of E-R diagram.
6. What is a relationship? Classify and give example.
7. Explain physical data independence.
8. Explain ISAM with example
9. Explain database users
10. Explain hierarchical data model.
11. Explain relational data model.
12. Explain outer Join with example.
13. List the components of data warehouse.

Five marks questions

1. Explain data processing cycle?
2. Explain various datatypes used in DBMS?
3. Explain Normalization with classifications and example
4. Explain cardinality with example.
5. Explain data independence in detail.
6. Discuss file organization with respect to physical data independence.
7. Explain the features of database system.
8. Explain DBMS Architecture.
9. Explain database model.
10. Explain Codd's rules for database management.
11. Write comparing RA and SQL.
12. List any five types of relational keys.
13. Explain Entity-Relationship in detail.
14. Explain the concept of Data abstraction.
15. Define and explain the phases of data mining.