

Srinivasan College of Arts & Science
(Affiliated to Bharathidasan University, Tiruchirappalli)
Perambalur – 621 212

COURSE MATERIAL

Database Systems
(16SCCCS4 / 16SCCIT9)



DEPARTMENT OF COMPUTER SCIENCE
& INFORMATION TECHNOLOGY

INTRODUCTION TO DBMS

System

A system is an integration of entities, alternatively designed as components, which have integration among them.

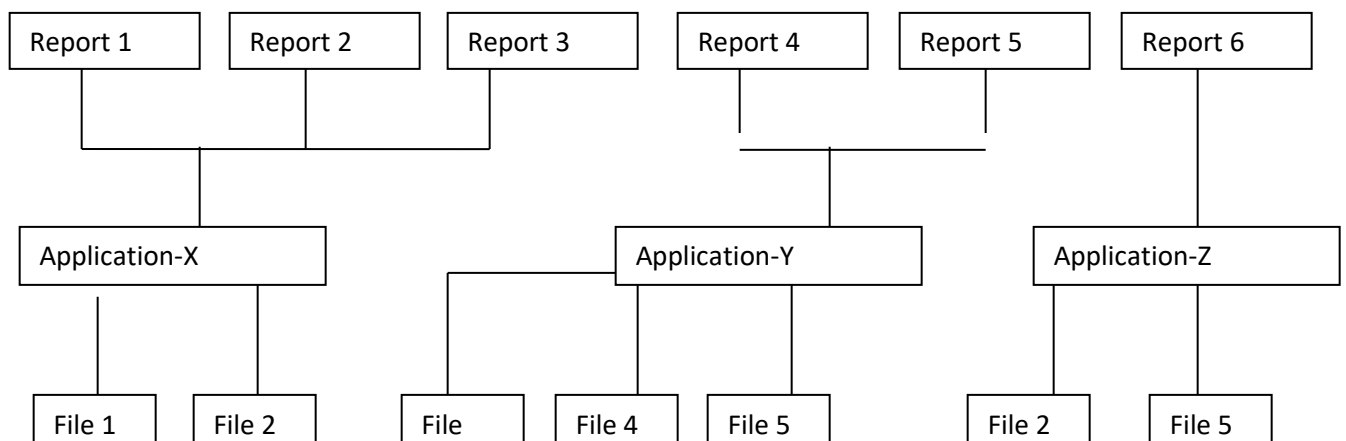
For E.g., consider a particular department in a college (or) university. The entities of the department are students, non-teaching staff, infrastructure etc. These entities interact with another.

The information system can be either a conventional file processing system (or) database management system.

Explain the Conventional File Processing System:

In the conventional File Processing System each and every sub system of the information system will have own set of files. As a result there will be a duplication of data between various sub systems.

The concept of Conventional File Processing System is shown below.

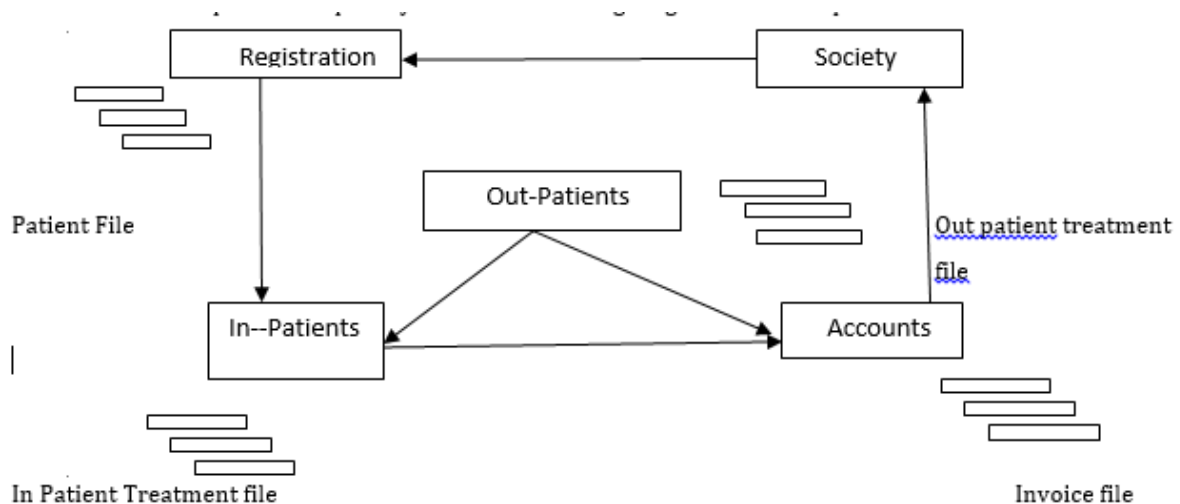


The above system consists of 3 sub-systems namely application-x, application-y and applications.

It is clearly that some of the files are duplicated in different subsystems of the conventional file processing system. This will intern increase the data redundancy.

Example of Conventional File Processing System:

Consider the example of a hospital system. The following diagram of the hospital is shown below.



The patients come to the hospital from the society. Upon the arrival of a preliminary registration is done by seeking information about the patient. Then depending on the type and illness, the patient will either be treated as out-patient (or) in-patient. In some cases initially a patient will be treated as out-patient and then the patient will be admitted as outpatient if necessary. Finally the bills are to be paid before the patient is discharged. In this system we are using four files. The files are

Patient File: At the Registration.

In-patient Treatment File: at the in-patient section.

Out-patient Treatment File: at the out-patient section.

In Voice File: at the accounts section.

The files are maintained in different sections of the hospital in a decentralized manner, certain data items (patient no, patient name, patient address) will be duplicated. This will have some undesirable results. Some of the difficulties of the conventional file processing system of the hospital are the following.

1. There may be a duplication of data in different sections of the hospital which would lead to inconsistency of data.
2. A lot of paper work and telephone calls would be required to synchronize file structure.
3. The system cannot provide answer to complex queries.

What are the Drawbacks of Conventional File Processing System ?

A list of drawbacks of the Conventional File Processing System is presented below.

1. Uncontrolled Redundancy of data.
2. Inconsistency of data.
3. Inflexibility
4. Lack of backup and recovery
5. Limited data sharing.
6. Poor enforcement of standards.
7. Unable to represent Relationships among data.
8. Excessive program maintenance and low programming productivity.

1. Uncontrolled Redundancy of data

Each sub system of an organization maintains own set of files without data sharing, the same data will be available in different files. This will result increased disc space, increased time of data entry and inconsistency of data.

2. Inconsistency of data

The uncontrolled redundancy of data will permit the system to have the same data in different files. As a result, a particular data element like patient name will be entered differently in different files. Which is nothing but inconsistency of that particular data element . while performing the basic data operations like updation, deletion, retrieval etc. This inconsistency will give misleading results.

3. Inflexibility

In the conventional file processing system generally top down approach will be followed in file design. In this approach a set of meaningful entities of the proposed system will be identified along with their attributes to create the files. The actual set of reports which is required for the system will not be considered on this approach. As

a result , it may not be possible to meet the requirements of the users fully. Also in the future , if there is a some changes in the user requirements , then the conventional file processing system will not be flexible to provide the results.

4. Lack of Backup and recovery

In this conventional file processing system there is no implicit facility for backup and recovery from system failure. It means that when an application program failed in middle of its work on its updating on a file.

5. Limited data sharing

In the conventional file processing system the data is stored in a decentralized manner, hence sharing of the data is complex one.

6. Poor enforcement of standards

Since different applications and their respective files were developed by different groups will design data fields, since each group will follow its own standards for fields name, fields width, fields type etc. This will create a serious difficulty while modifying programs and data structures by different groups of users which will leads to low programmer productivity.

7. Unable to represent relationships among data.

In the conventional file processing system there is no implicit facility to represent relationship among data in different file for a single system.

8. Excessive program maintenance & Low program productivity

Since the different applications are developed differently by different groups in terms of file specifications and program specifications, it will be very difficult to modify the programs and data structure at a later stage by a different group. Many of program variables may be defined differently in different programs. All these difficulties will leads to excessive maintenance.

Programmer productivity is a measure of time taken to develop an application. If the time taken to develop an application is low then the programmer productivity is high and vice versa.

What is DATABASE?

- Database is a collection of related data (or) files.
- Data means known facts that can be recorded and that have implicit meaning.

For E.g., consider the names, telephone numbers and addresses of the people. We have recorded this data in an address book (or) we may have stored it on a file in the hard disk, using a computer and software such as Microsoft Excel (or) MS Access. This collection of inter related data is a database. Generally a database contains one data file (or) large number of data files. The database is organized in such a way that a application programs will quickly retrieve the required data.

What is Database Management System (DBMS)?

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is general-purpose software system that facilitates the defining, constructing, manipulating and sharing database among various users and applications.

Defining a database involves specifying the data types, structures, and constraints for the data to be stored in the database.

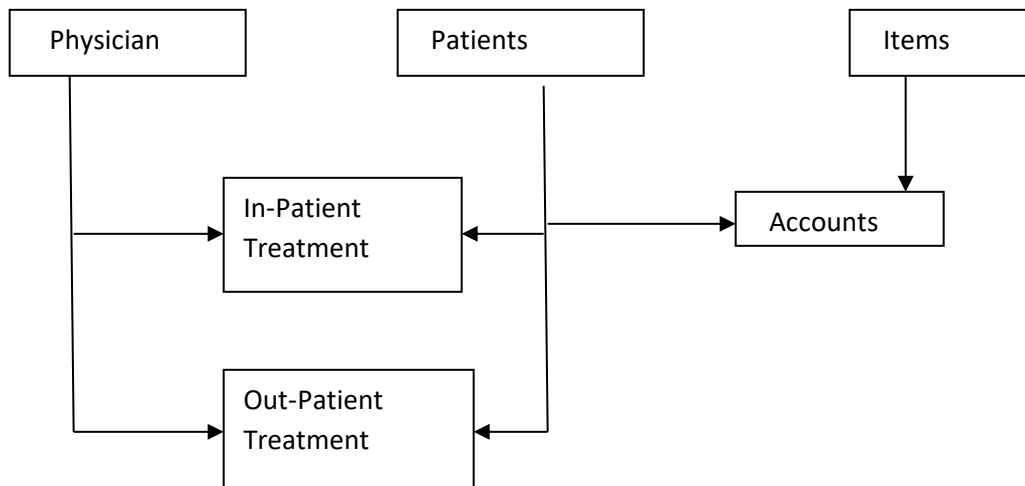
Constructing the database is the process of storing the data itself on some storage medium that is controlled by the DBMS.

Manipulating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the database, and generating reports from the data.

Sharing a database allows multiple users and programs to access the database simultaneously. Other important function provided by the DBMS is unauthorized users cannot access the database.

Example of Database Management System

Consider the example of the hospital system which deals in-patients as well as out- patients. The hospital system is shown below.



In the Conventional File Processing System there will be a separate system for in-patients, Out-patient and Accounts. But under the database approach, all the necessary files are included in a single database, which minimizes redundancy of data and facilitates sharing. As a result all the drawbacks of Conventional File Processing System are eliminated in the Database System.

What are the Advantages of Database Management System:

The Advantages of Database Management System are

1. Controlled Redundancy of data.
2. Consistency of data.
3. Flexibility
4. Backup and recovery
5. Enhanced Data sharing.
6. Better enforcement of standards.
7. Relationships among data.
8. Reduced program maintenance and Increased programmer productivity.

1. Controlled Redundancy of data

The database approach uses a centralized place to save data hence the amount of data redundancy will be minimized. Data redundancy can be minimized by applying normalization process in database design.

2. Consistency of data

The controlled redundancy will minimize the presence of same data in different files. This will lead consistency of data by reducing multiple inserts and updates on same data at different places.

3. Flexibility

In database approach , the database design based on bottom up approach. In this approach all the reports that are currently used and various expected reports are taken into design the database. When some changes in report requirements occurs revision of database can be done with minor changes in the database.

4. Backup and Recovery

In DBMS if a transaction fails in middle of its execution due to system failure then DBMS will get back the data into its consistent state as before.

5. Enhanced Data Sharing

In database approach uses centralized database hence same data can be shared by different applications or users simultaneously with concurrency control. In DBMS multiple users will access same data and can do changes.

6. Better enforcement of standards

Since different files of the database are design at a time of a different subsystems, it will be better enforcement of standards in terms of defining field name, field width , field type etc.,

7. Relationships among data

We can apply relationships among data to improve performance of applications and consistency (correctness) of data. For example foreign key constraint makes a relationship between Employee and Department information.

8. Reduced program maintenance and increase of programming productivity.

Different applications are developed under the coordination of the database administrator. As a result, there will be an integrated effort among the different groups in terms of file design and program design. This will reduce the program maintenance.

Programmer productivity is a measure of time taken to develop an application. In the database approach, data is separated from programs. There are many fourth-generation languages available to access and manipulate the database.

Because of the advanced capabilities of fourth-generation languages, the time taken to develop an application will be less when compared to the time taken to develop using a conventional file processing system.

What are the Applications of DBMS ?

The applications of a database are:

1. Banking: For customer information, accounts, loans, Bank transactions.
2. Airlines: For reservations and scheduled information.
3. Universities: For student information, course registration and grading.
4. Telecommunication: For keeping records of call mode, generating monthly bills, maintaining balances on pre-paid cards and storing information about the communication network.

What are the Disadvantages of DBMS ?

The following are the disadvantages of using DBMS.

1. Increased complexity.
2. Requirement of new and specialized manpower.
3. Large size of DBMS.
4. Increased installation and maintenance cost.
5. Conversion cost.

1. Increased complexity

A multi user DBMS becomes an complex piece of software due to expected functionalities from it. It becomes necessary for database designers, developers, database administrator and end users to understand these functionalities. Failure to understand that can lead to bad designed decisions.

2. Requirement of new and specialized manpower

Because of rapid in database technology and organizations, business need to trained manpower on regular basis to design and implement of database administrative services and manage a staff of new people, therefore an organization needs to maintain specialized skill person.

3. Large size of DBMS

The DBMS occupies many Giga Bytes of storage space and requires more amount of main memory to run efficiently.

4. Increased installation and maintenance cost

The DBMS software has a high initial cost. It requires trained person to install and operate. and also has more annual maintenance. Installing such software's also requires upgrades to the hardware and software.

5. Conversion cost

The conversion cost from old database technology to modern database environment is high.

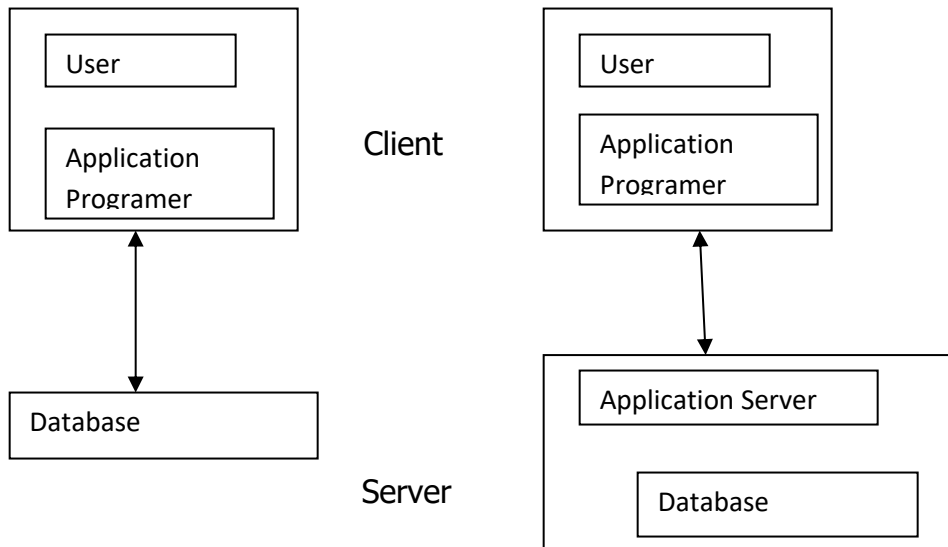
Explain the DBMS Architecture

1. The main aim of database system is to provide an abstract view of data hiding of certain detail of how data is stored. And manipulated, to satisfy these needs to develop architecture for database system.

2. In early days the whole DBMS package was a single package where as modern DBMS is based on client-server architecture.

3. Under the client-server architecture the database is not present in the client machine. But the client machine connected to the database system through Network and server.

4. There are two types of DBMS architecture as shown below.



Two Tire Architecture

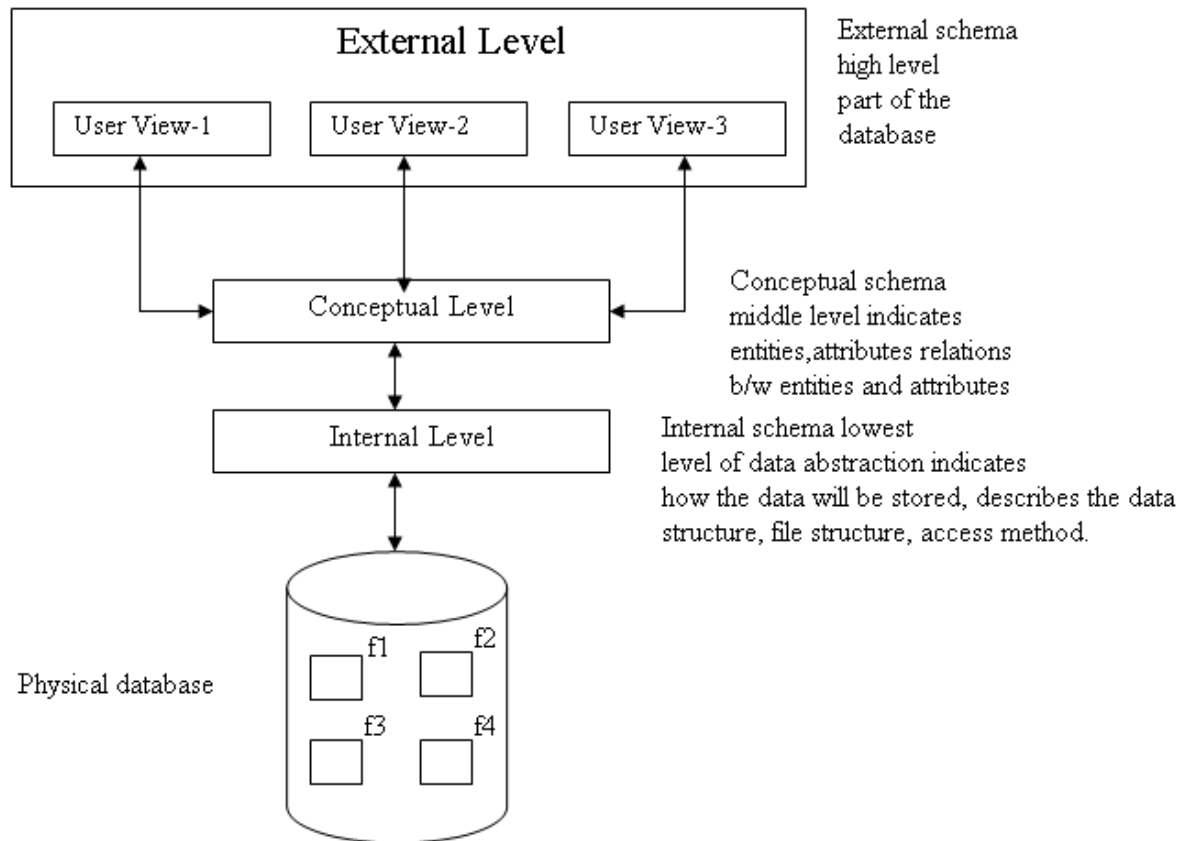
Three Tire Architecture

In two-tier architecture, the application is a component that resides on the client machines that communicate server machine through query language statements.

In three-tier architecture, the client machine cannot directly communicate with an application server, the application server communicates with a database to access data. Three tier architectures are more suitable for large applications.

Explain the Database Architecture

A database Architecture is shown below depending upon the three tier architecture. It contains of 3 levels.



Internal level

The internal schema defines the internal level. The internal level is the lowest level of data abstraction. This level indicates how the data will be stored into the database and describes the file structures and data structures and methods to be used by the data base.

Conceptual level

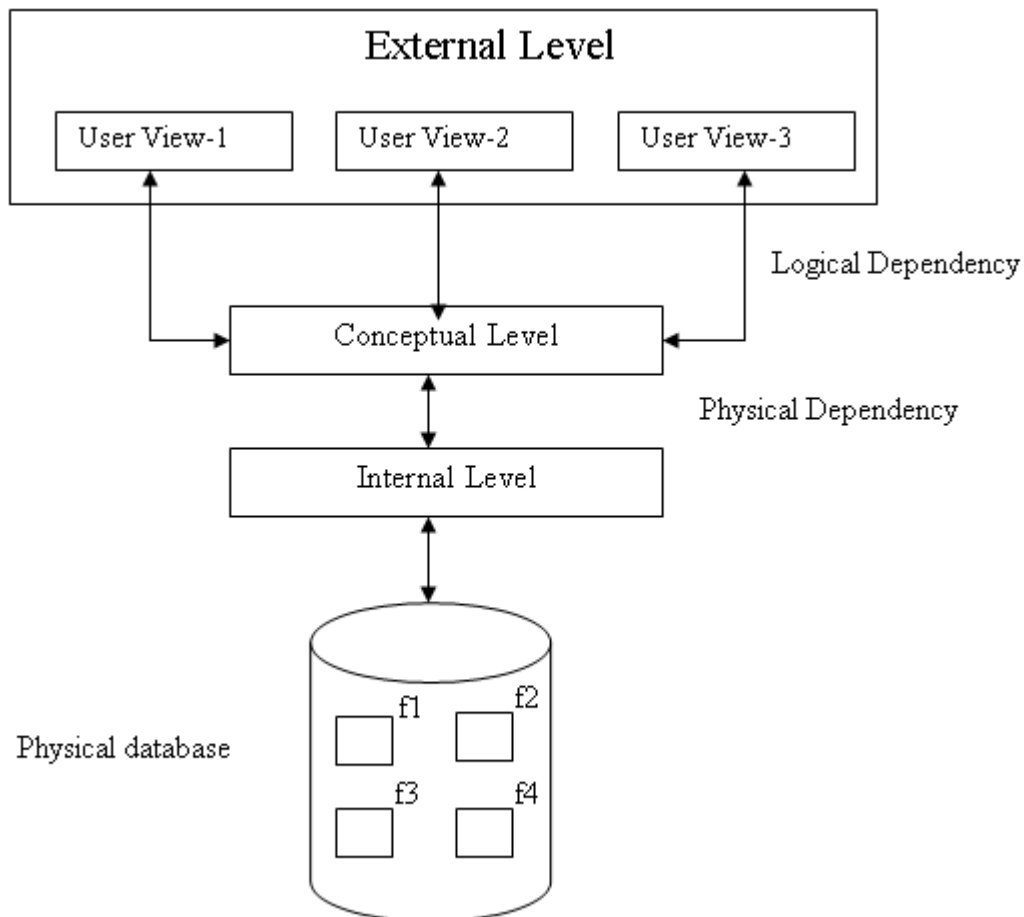
The conceptual schema defines the conceptual level. The conceptual level is the middle level abstraction. This level indicates entities, attributes, relationship between entities and attributes.

External level

External schema defines the external level. The external level is the highest level of data abstraction. This level describes part of database. i.e., relevant to the user.

What is Data independency ?

Data independency is the characteristic of database. To change the schema at one level without changing the schema at the higher level. There are two types of data independency as shown below.



Physical independency

In physical independency, changes to the internal schema such as file structures, accessing methods and devices used for store the data must be possible, without changing the conceptual schema and external schema.

Logical independency

In logical independency, changes to the conceptual schema such as addition and deletion of entities, addition and deletion of attributes, addition and deletion of relationships must be possible without change in external schema.

What are the Functions or services of DBMS ?

The functions and services of DBMS are

1. Data storage Management

DBMS creates the structure for database in the physical storage devices. It provides a mechanism for permanent storage of data.

2. Data Manipulation Management

The DBMS provides ability to add new data into the database (or) retrieve, update and delete existing data in the database.

3. Data Definition Management

The DBMS creates the structure of data in which the data is stored.

4. Data dictionary

The DBMS provides a data dictionary in which stores the description of data items.

5. Authorization

The DBMS protects the database against unauthorized access either intentional (or) accidental.

6. Backup and recovery

The DBMS provides a mechanism for Backup data periodically and recovery from different types of failures.

7. Concurrency control

The DBMS supports sharing of data among multiple users. The DBMS provides a mechanism for concurrent access to the database.

8. Transaction Management

The transaction is a series of database operations, which access (or) changes the content of the database. This is done by the transaction management.

9. Data Independency Service

The DBMS supports the independency of the programs from its structure of the database.

10. Integrity Service

The DBMS provides integrity service to store the data into the database (or) to change the data into the database follows certain rules.

What are the Functions database Administrator ?

Database administrator is an individual person with an overview of one (or) more databases and also controls the design and use of database.

Functions and responsibilities of DBA are

1. Defining conceptual schema and database creation

The DBA creates the conceptual schema such as defining entities and attributes, deleting entities and attributes and modifying entities and attributes etc. The DBA also creates the structure of the database.

2. storage structure and access method definition

The DBA defines the storage structure of the data. And access methods of the database.

3. Granting authorization to the user

The DBA grants the access to use the database to its users. The authorization information is kept in a system, the database system consults whenever someone attempt to access the data in the system.

4. Routine maintenance

The DBA maintains periodically backups of the database either on hard disc (or) CD to prevent loss of data in case of failure.

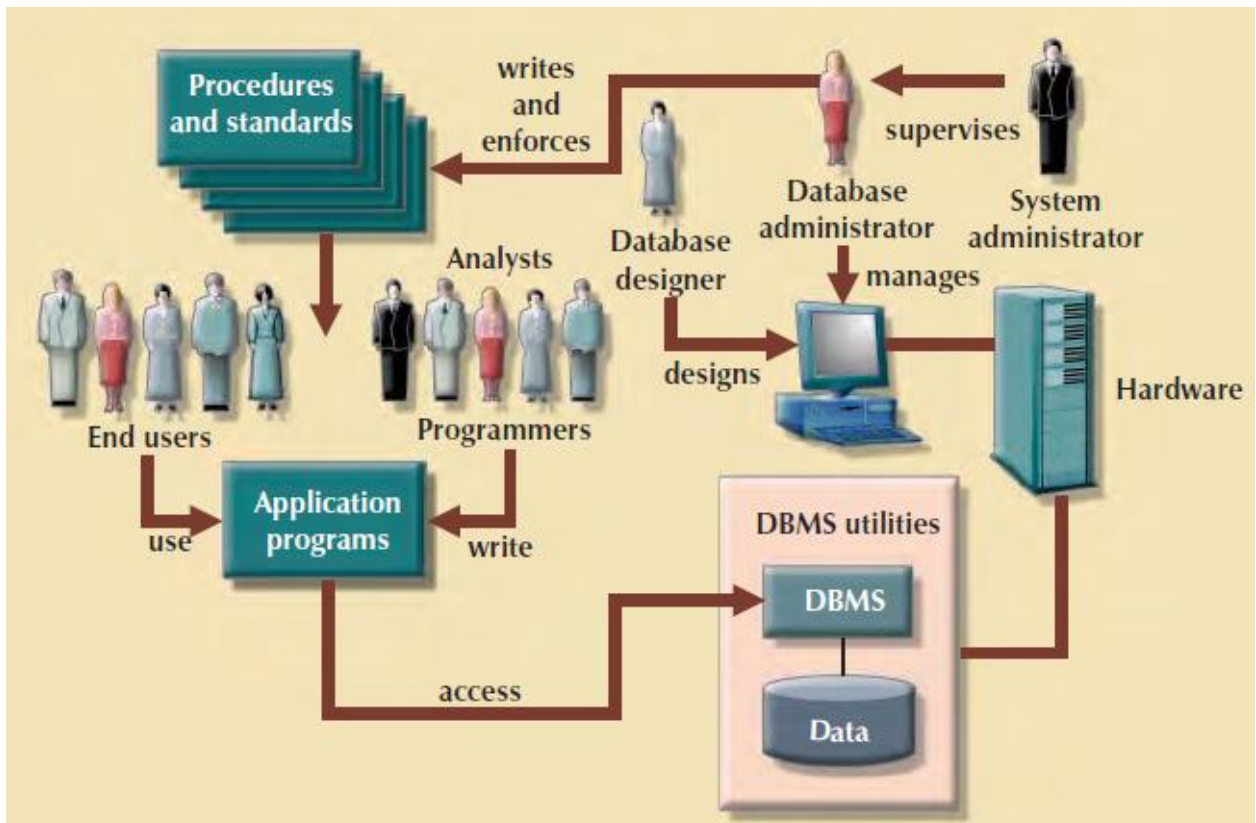
5. Job monitoring

The DBA is responsible for the performance of data is not decreased.

What are the various components of Database Systems ?

The database system is composed of the five major components.

1. Hardware
2. Software
3. People
4. Producers
5. Data



1. **Hardware:** Hardware refers to physical components of the system.
E.g. storage devices, printers etc.
2. **Software:** software is a set of programs. To make database system function fully, three types of software's are needed. They are
 - a. Operating System Software
 - b. DBMS software.
 - c. Application Programs and utility software.
 - a. **Operating system software:** operating system manages all hardware components and run other software's on the computer.
E.g. WINDOWS, LINUX
 - b. **DBMS Software:** manages the database within the database system.
E.g. Oracle, SQL, MS Access

C. Application programs and utility software: Application programs are used to access and manipulate data to generate reports and making decisions.

Utilities are the software tools used to help, manage the database systems computer components.

- 3. People:** There are five types of users in the database system.
 - a. System Administrator:** to see the database systems general operation.
 - b. Database Administrator:** see the database is functionality properly.
 - c. Database designers:** design the database structure.
 - d. System analyst and programmers:** Design and implement the application programs.
 - e. End user:** use the application programs to run the organization daily operations.
- 4. Procedures:** procedures are the set of rules based on design and use the database.
- 5. Data:** The data is a facts stored in the database. Because data are the raw material from which the information is generated.

What is Data?

A system consists of interrelated entities, each entity has a set of attributes of entities of the system.

What is Information ?

Information is nothing but processed data.

Define Meta data

Meta data is the data about the data i.e., information for accessing the data.

Explain the Terminology of a file

Field: A field is the lowest level of data item of an entity which is alternatively called as an attribute of that entity.

Emp → Empno Empname Empaddress

Record: Record is the collection of fields (or) attributes of an entity.

Empno	Empname	Empaddress
1	sweaty	banglore

File: File is a collection of records having same set of fields arranged in the same sequence.

Empno	Empname	Empaddress
1	manimala	Chennai
2	priya	Hyderabad

Key field (or) Primary key: A key field is said to be key field (Or) primary key if it can identify a record uniquely in a file.

e.g. student no in student file

emp no in emp file.

Non key field (or) secondary key: A field is said to be Non key field (or) secondary key if it cannot identify a record uniquely in a file.

e.g. student name in student file.

Emp name in emp file.

Schema: it is a overall view of all the files in the database.

Subschema: A portion of the schema of a database is called as subschema.

Data Models

Data Modeling and Data Models:

Data model:

A data model is a collection of concepts that can be used to describe the structure of a database

Data modeling in the first step in designing a database refers to the process of creating a specific data model for a problem.

A model is an abstraction of a real world object. A data model represents data structures and their characteristics, relations, constraints and transactions.

Data model is an iterative process we start with a simple understanding of the problem increases, and finally design a database in a specific database model.

Importance of Data Models:

1. Data Model can facilitate interaction among the designer, the application programmer and the end user.
2. Applications are used to transform data into information. But data are viewed in different ways by different people.
3. For e.g. the manager and clerk both are working in the same company, the manager have wide view of company data than the clerk.
4. A company president has universal view of data.
5. Different Managers views data differently in a company. The inventory manager is more concerned about inventory levels, while purchasing manager concerned about items and supplies.
6. Application programmers have another view of data i.e., concerned with data locations and formatting.
7. A house is a collection of rooms, if someone is going to build a house, they have the overall view i.e., provided by blue print. A sound data environment requires an overall database blue print based on appropriate data model.
8. When a good database blue print is available, an application programmer view of data is different from the managers and end users. When a good database blue print is not available problems are likely to ensure.

Data Model basic building blocks:

The basic building blocks of data models are entities, attributes, relationships and constraints. An entity represents a real world object person (or) place.

For e.g., a customer entity have different of customers.

An attribute is a characteristic of an entity.

For e.g. customer entity have attributes customer_no, customer_name, customer_address etc. A relationship describes an association between entities. Data models use three types of associations. One-to-many, many-to-many and one-to-one.

One-to-many (1:M, 1.....*): A painter paints many different paintings. Therefore, the database designer label the relationship PAINTER PAINTS PAINTINGS as one-to-many.

Many-to-many (M:N, *.....*): An employee may learn many job skills and each job skill may be learned by many employees. Therefore, the database designer label the relationship Employee learns skills as many-to-many (M:N).

One-to-one (1:1, 1.....1): Each store manager manages only a single store. Therefore, the data designer label the relationship employee manages stores as one-to-one (1:1).

By using the business rules we can properly identify entities, attributes, relationships and constraints.

BUSINESS RULES:

A business rule is a description of a policy, procedure (Or) principle within a business organization. Examples of business rule.

1. A customer may generate many invoices.
2. A training session cannot be scheduled for fever than 10 employees or for more than 30 employees. These business rules establish entities, relationships and constraints.

The first business rule establishes two entities (customer, invoices) and a one-to-many relationship between these two entities.

The 2nd business rule establishes a constraint. (No fewer than 10 people (or) more than 30 people) and two entities (training, people) and a relationship between employee and training.

DISCOVERING BUSINESS RULES:

The main source of business rules are company manager, policy manager, department manager and written documents such as company's procedures, standards (or) operation manuals. A faster direct source of business rules is direct interviews with the concerned persons.

Translating business rules into Data Model:

General rule: A noun in a business rule be translate into an entity in that model, and a verb associating nouns will translated into a relationship among the entities.

For e.g. the business rule " customer may generate many voices "

Containing two nouns (customer and invoices) and a verb (generate) that associates the noun.

To proper identify the type of relationship, the relationships are bi-directional.

For e.g. the business rule " A customer may generate many invoices", the relationship is one-to-many (1:M, 1.....*). Customer is the 1 side and invoice is the many side.

Evolution of Data Models:

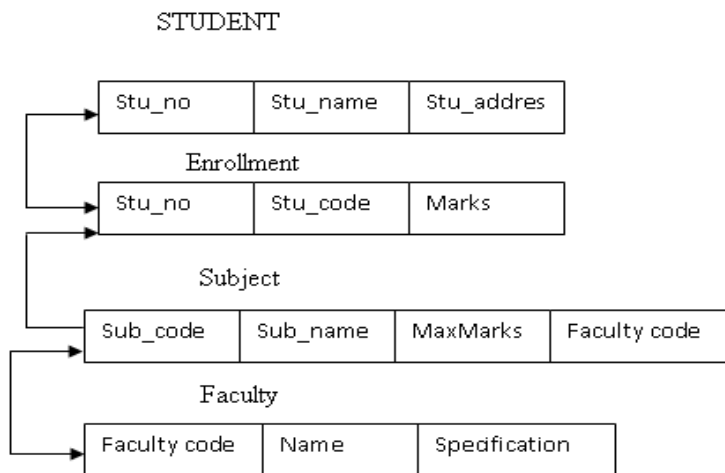
Generation	Time	Model	Example
First	1960-70	File system	VMS
Second	1970	Hierarchical and network data models.	IMS, focus IDMS

Third	Mid 1970's-present	Relational data model	M.S.Acess, Oracle
Fourth	Mid 1980's-present	Object Oriented Model Extended Relational Model	Versant Objectivity
Fifth	Present-Future	XML	Oracle log

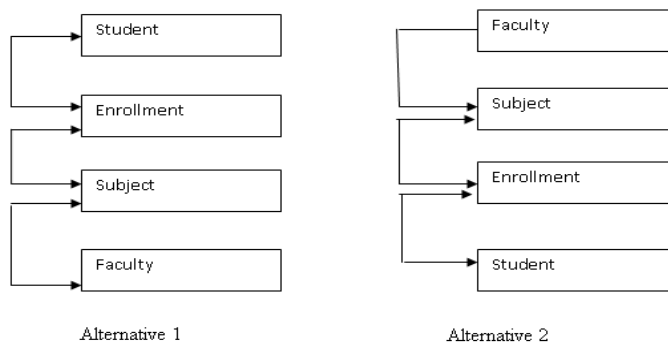
HIERARCHICAL DATA MODEL:

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**.

1. The hierarchical data model consists of a set of nested relationships one-to-many and one-to-one association.
2. In hierarchical data model the relations are presented in the form of tree-structure in which the root segment is kept at the top and further branches emanate downwards from the root segment.
3. In this model the type of association can be one-to-one and one-to-many. This means that many-to-one association is not permitted. This is equivalent to say that multiple percentages for a child segment is not permitted.



The above conceptual data model can be mapped into any one ways as shown below.



An alternative1 student file is kept at the root segment of the tree and the faculty file is kept at the bottom of the tree. By mapping the conceptual data model into the hierarchical data model the following facts are observed.

1. The association from student to enrollment is one-to-many. This mapped without any modifications.
2. The association from enrollment to subject is many-to-one. This is not permitted in hierarchical data model. Hence it is modified into one-to-one association.
3. The association from subject to faculty in many-to-one. This is not permitted in hierarchical data model. Hence it is modified into one-to-one association.

In alternative1 while mapping the conceptual data model into hierarchical data model, the many-to-one association presents at two levels are modified into one-to-one association. These modifications will increase the data redundancy.

In alternative2 the faculty file is kept at the root of the tree and student file is kept at the bottom of the tree. While mapping the conceptual data model the following facts are observed.

1. The association from faculty to subject file is one-to-many. So it is mapped without any modifications.
2. The association from to subject enrollment is many-to-one. This is not permitted in hierarchical data model. Hence it is modified into one-to-one association.
3. The association from enrollment to student is many-to one. This is not permitted in hierarchical data model. Hence it is modified into one-to-one association.

Finally which alternative has less redundancy should be selected for implementation.

In alternative2, the association change between enrollment and student. That means we are changing one type. When we compare alternative2 with alternative1, alternative2 has less redundancy and it is implemented.

Advantages:

1. It promotes data sharing.
2. Parent/Child relationship promotes conceptual simplicity.
3. Database security is provided and enforced by DBMS.
4. Parent/Child relationship promotes data integrity.
5. It is efficient with 1:M relationships.

Disadvantages:

1. Complex implementation requires knowledge of physical data storage characteristics.
2. Changes in structure require changes in all application programs.
3. There are implementation limitations (no multi parent or M:N relationships).
4. There is no data definition or data manipulation language in the DBMS.
5. There is a lack of standards.

NETWORK DATA MODEL:

A Network data model consists of a set of pair wise association between the entities.

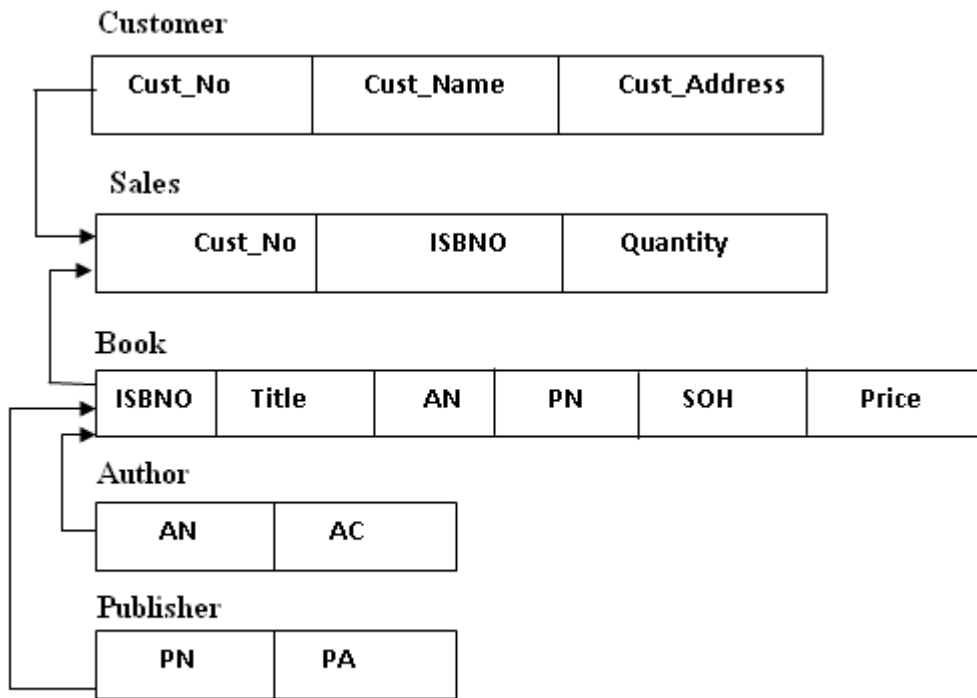
The Network data model was created to improve database performance, database standards and represent complex relationships effectively than the hierarchical data model.

To establish database standards, the conference of database system languages (CODASYL) created the database task group (DBTG). The DBTG define standard specifications for database creation and data manipulations.

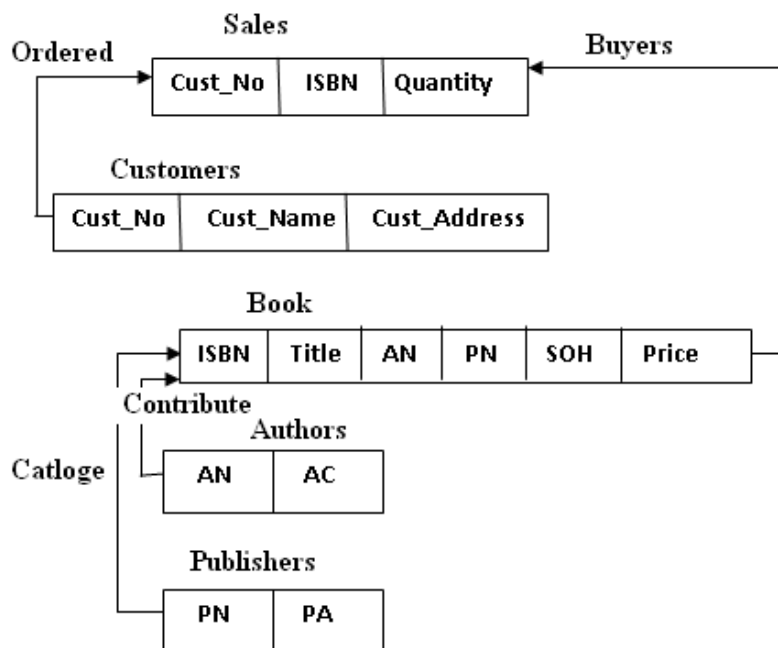
1. **Schema:** The schema provides overall view of the database to the administrator.
2. **Sub Schema:** The sub schema which defines the portion of the database seen by the application programs.
3. **Database Management Language:** That defines the environment in which data can be changed. The DBTG specify 3 DML components.
 - a. A schema data Definition Language (DDL), which enables the data base administrator to create the database.
 - b. A subschema DDL, which allows the application programs to define database component that will be used by the application.
 - c. A Data Manipulation Language, to manipulate the data in the database.

In Network data model, the Network database as a collection of records in one-to-many record to have more than one parent.

In Network database, a relationship is called a set. Each set contains two entities one entity is owner and other entity is member.



Conceptual Data Model for Book House



Sets

Set Name	Owner	Member
Ordered	Customer	Sales
Buyers	Book	Sales
Contribute	Author	Book
Catalog	Publisher	Book

Advantages:

1. Conceptual simplicity is at least equal to that of the hierarchical model.
2. It handles more relationship types, such as M:N and multi-parent.
3. Data access is more flexible than in hierarchical and file system models.
4. Data Owner/Member relationship promotes data integrity.
5. There is conformance to standards.
6. It includes data definition language (DDL) and data manipulation language (DML) in DBMS

Disadvantages:

1. System complexity limits efficiency—still a navigational system.
2. Navigational system yields complex implementation, application development, and management.
3. Structural changes require changes in all application programs.

RELATIONAL DATA MODEL:

The relational model was introduced in 1970 by E.M.Codd. The foundation of relation is a mathematical concept known as relation. The Relation is composed of intersecting rows and columns. Each row in a relation represents a tuple. Each column represents an attribute.

The relational data model is implemented through a Relational Database Management System (RDBMS).

Tables are related through the sharing of common attribute. For e.g. the table agent and customer as shown below.

Agent

Agent_Code	Agent_Name	Agent_Address	Agent_PhoneNo	Agent_Area code

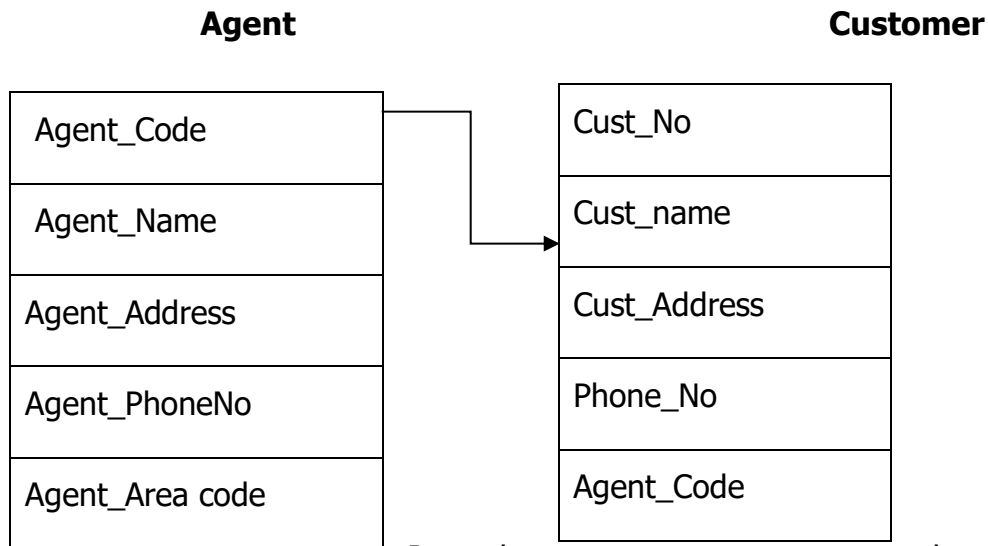
Customer

Cust_No	Cust_name	Cust_Address	Phone_No	Agent_Code

By matching the Agent_Code in the customer table with Agent_Code in the Agent table we can find Agent details of that customer.

The relationship types one-to-one, one-to-many and many-to-many have in a relational data model.

A relationship diagram is a representation of entities, the attributes within the entities and the relationship between the entities.



In the above diagram, the relationship is one-to-many. The symbol ∞ indicates many.

The customer represents "many" sides, because an AGENT can have many CUSTOMERS.

The AGENT represents the "1" side because each CUSTOMER has only one AGENT.

The languages which are supported to relational data model is powerful and flexible. Because of that the relational data model is popular. FoxPro, database, M.S.Acess, SQL are relational database software's. This software's allows the user to specify what must be done without specifying how it must be done.

SQL based database applications involves 3 parts. 1. End user interface. 2. Set of tables stored in the database. 3. SQL Engine.

Advantages:

1. Structural independence is promoted by the use of independent tables. Changes in a table's structure do not affect data access or application programs.
2. Tabular view substantially improves conceptual simplicity, thereby promoting easier database design, implementation, management, and use.
3. Ad hoc query capability is based on SQL.
4. Powerful RDBMS isolates the end user from physical-level details and improves implementation and management simplicity.

Disadvantages:

1. The RDBMS requires substantial hardware and system software overhead.
2. Conceptual simplicity gives relatively untrained people to use a good system poorly, and if unchecked, it may produce the same data anomalies found in file systems.
3. It may promote "islands of information" problems as individuals and departments can easily develop their own applications.

ENTITY RELATION MODEL:

Peter Chen first introduced the E.R.data model in 1976; it was the graphical representation of entities and their relationship in a database.

E.R. models are normally represented in an entity relationship diagram.

The E.R.Model is based on the following components.

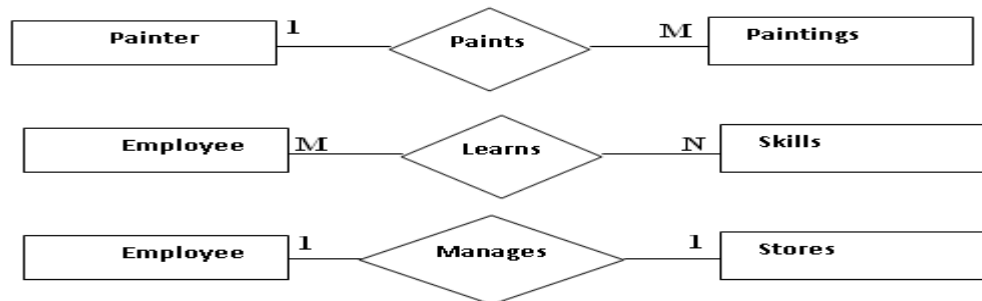
- a. **Entity:** entities are the real time objects. Entities represented by a rectangle.
e.g. painter, employee, skills, noun.
- b. **Attribute:** Attributes are the characteristics of entities.
e.g. Empno, Empname, Empaddress etc.
- c. **Relationships:** A relationship describes association among the entities. There are three types of relationships, one-to-many, many-to-many and one-to-one.

There are two types of ER notations.

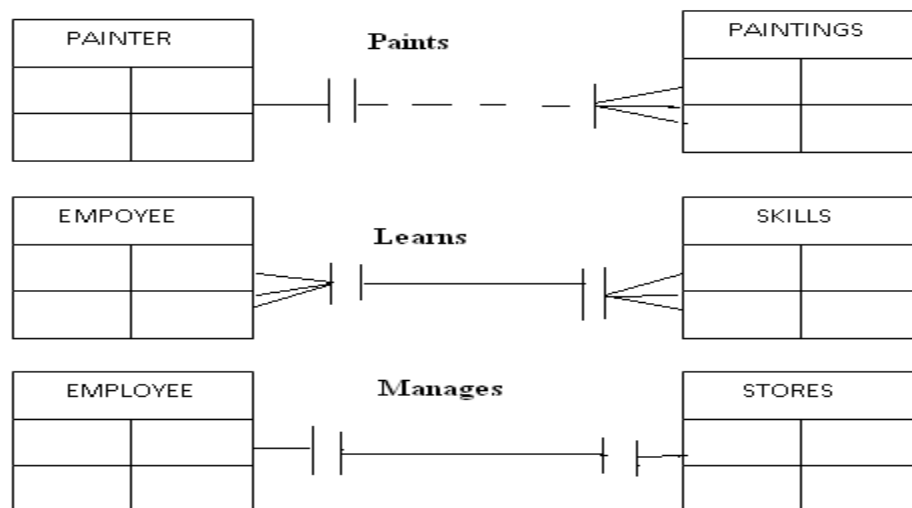
1. Chen notation
2. Crow's foot notation.

For different types of relationships.

CHEN NOTATION



Crown's Foot Notation



In Chen notation, entities are represented rectangle and entity names are written in the capital letters at the centre of the rectangle. Relationships are represented by a diamond. The diamonds are connected to entities through a relationship name is written inside the diamond.

In the crows foot notation, the crow foot is derived from the three pronged symbol used to represent many relationships. In this notation, the one represented by a short line segments, and many is represented by the crow's foot. The relationship name is written above the relationship line. The relationships are also show in vertical.

Advantages:

1. Visual modeling yields exceptional conceptual simplicity.
2. Visual representation makes it an effective communication tool.
3. It is integrated with dominant relational model.

Disadvantages:

1. There is limited constraint representation.
2. There is limited relationship representation.
3. There is no data manipulation language.
4. Loss of information content occurs when attributes are removed from entities to avoid crowded displays. (This limitation has been addressed in subsequent graphical versions)

OBJECT ORIENTED MODEL:

In the Object Oriented Data Model (OODM) both data and their relationships are contained in a single structure known as an Object.

Object Oriented Data Model has allowed an object, the object contains operations that can be performed on it, such as changing data values, finding a specific data value, and printing data values.

The OODM is based on the following components.

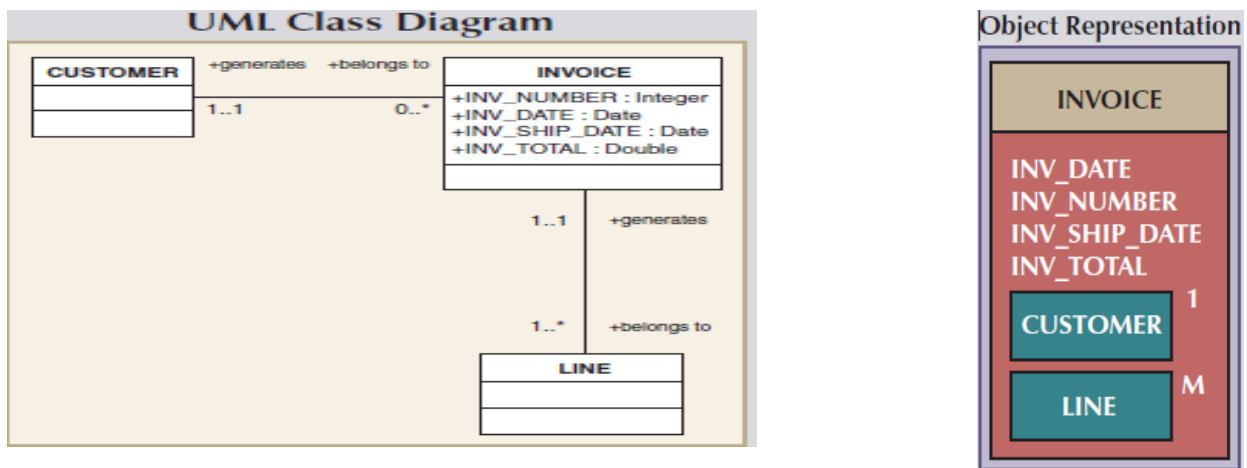
- a. An object is an abstraction of a real world entity.
- b. Attributes describes the properties of an object.
E.g. Person object contains the attribute name, social security number, date of birth etc.
- c. A collection of similar objects contains attributes and methods. By using those methods change data values, find data values and print data values in the objects.
- d. Classes are organized in a class hierarchic key. The class hierarchic key is similar to upside down tree. In which each class has only one parent.
- e. One of the properties of object oriented data model is inheritance. By using the inheritance we can inherit attributes and methods from super class to sub classes.

E.g. customer and employee sub classes of the person super class. Customer and employee will inherit all attributes and methods from person.

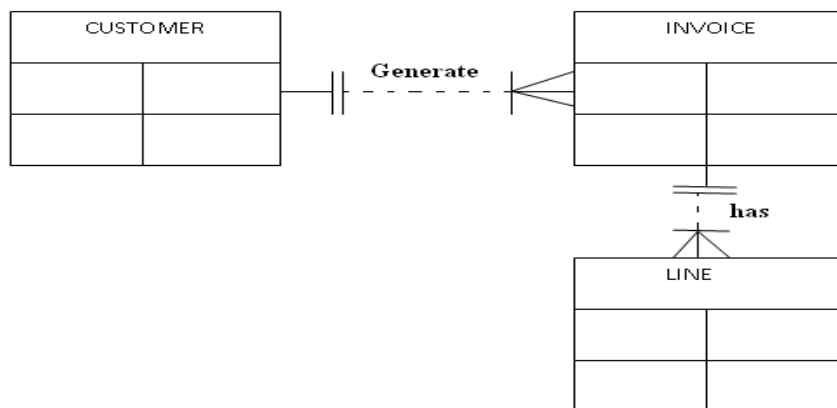
- f. Object oriented data models are drawn using unified modeling language (UML) class diagram. The UML class diagrams are used to represent data and other their relationships.

For E.g. Let us use invoice program. In this case the invoices are generated by customer, each invoice contains one (or) more lines, and each line represents an item purchased by customer.

The following diagram shows an object representation UML class diagram, ER model for invoice.



ER Diagrams



The object representation of invoice includes all the related objects within the same object. The 1 next to the customer object indicates that each invoice related to

one customer. The M next to the line object indicates that each invoice contains no. of lines.

The UML class diagram uses 3 separate classes (customer, invoice and line) and two relationships to represent this problem.

The E.R Model also uses the 3 separate entities and two relationships to represent the invoice problem.

Advantages:

1. Semantic content is added.
2. Visual representation includes semantic content.
3. Inheritance promotes data integrity.

Disadvantages:

1. Slow development of standards caused vendors to supply their own enhancements, thus eliminating a widely accepted standard.
2. It is a complex navigational system.
3. There is a steep learning curve.
4. High system overhead slows transactions

Relational Database Model

In Relational Data base model records are stored into tables. Relational data model is easier to understand than the hierarchal data models and network data models. Relational data model provides a logical view of the data and its relationship among other data.

Tables and Characteristics:

A Table is composed of rows and columns. Each row of the table is called as tuple. Each column of the table is called as attribute. A table is also called as Relation. A table contains a group of related entities.

Characteristics of a Table:

1. A table is composed of rows and columns.
2. Each row of the table represents one entity (tuple) in the entity set.
3. Each column represents an attribute and each column has distinct name.
4. Each cell represents a single value.
5. All values in a column must have same data format.
6. Each column has a specified range of values which is called domain.
7. The order of the rows and columns is immaterial to the DBMS.
8. Each table must have an attribute or group of attributes that uniquely identified each row.

The following Student table shows above characteristics.

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS
321452	Bowser	William	C	12-Feb-1975	42	So
324257	Smithson	Anne	K	15-Nov-1981	81	Jr
324258	Brewer	Juliette		23-Aug-1969	36	So
324269	Oblonski	Walter	H	16-Sep-1976	66	Jr
324273	Smith	John	D	30-Dec-1958	102	Sr
324274	Katinga	Raphael	P	21-Oct-1979	114	Sr
324291	Robertson	Gerald	T	08-Apr-1973	120	Sr
324299	Smith	John	B	30-Nov-1986	15	Fr

1. The student is composed of 8 tuples (rows) and 6 attributes (columns).
2. In the student table the primary key is STU_NUM(student number) by using this attribute we can identify a record uniquely in a student table.

KEYS**Key field or Primary Field:**

A key consists of one or more attributes that determines other attributes. For example, an invoice number identifies attributes such as invoice date, customer details, items details and amount.

The statement "A DETERMINES B" indicates that if we know the value of attribute A then determine the value of B.

Eg: In the student table if we know the value of student_number then determines student last name, st_fname, stu_initial. This can be represented in the following way.

STU_NUM → STU_LNAME, STU_FNAME, STU_INIT.

Functional Dependency:

The term Functional Dependency can be defined for "A DETERMINES B", if each value in a column "A" determines only one value in column B.

Eg: STU_NUM functionally determines STU_LNAME (or STU_LNAME not functionally depends on STU_NUM).

STU_NUM → STU_LNAME.

Composite Key or Fully Functional Dependency:

A key may be composed of more than one attribute; such a multi attribute key is known as Composite key.

Eg: STU_LNAME, STU_FNAME, STU_INIT, PHONE_NO → STU_CLASS, STU_HOURS.

The combination of STU_LNAME, STU_FNAME, STU_INIT, PHONE_NO can determine the STU_CLASS, STU_HOURS. If the attribute "B" is functionally depends on "A". composite key "A", but not on any subset of composite key. The attribute "B" is fully functionally depends on "A".

Super Key :A sub set of Attributes that uniquely identifies a tuple(row) in a relation(table).

Eno	Ename	Salary	Dept_no	Voter_Id
1	Raju	20000	10	V12345
2	Prasad	40000	10	V12222
3	Raju	20000	20	V45666

{Eno } :No two rows have same Eno (Eno uniquely identifies a tuple(row) in a relation)

{Ename } : Two employee's may have same name.

{Voter_id} :No two rows have same Voter_id (Voter_id uniquely identifies a tuple(row) in a relation)

{Eno, Ename } : Eno itself uniquely identifies a tuple(row) in a relation, hence combination of Eno and Ename also uniquely identifies a tuple(row) in a relation

Eg : In a STUDENT table

STUDENT{STU_NUM,STU_LNAME,STU_FNAME,STU_INIT,STU_DOB,STU_HRS,STU_CLASS}

Super Keys can be identified as fallows.

- {STU_NUM}
- {STU_NUM, STU_LNAME}
- {STU_NUM, STU_LNAME, STU_INIT}

*Minimal Super Key (Key) :

Definition : A Minimal Super Key (Key) K is a superkey with the additional property that removal of any attribute from K will cause K not to be a superkey any more.

{Eno } : is Minimal Super Key (A Super Key which have only one attribute is Minimal Super Key)

{Voter_id} : is Minimal Super Key

{Eno, Ename} : is Not a Minimal Super Key (Removal of Ename from {Eno, Ename} = {Eno} is also a Super Key } hence {Eno, Ename} is not Minimal Super Key.

Candidate Key :

Definition : If a relation schema has more than one key (Minimal Super Key) then each of them is called as candidate key.

- **One of the candidate keys is arbitrarily designated to be the primary key, and the others are called secondary keys(or Alternative key).**
- A key formed by combining at least two or more columns is called composite key

Primary Key :

Definition : Set of attributes of a relation which uniquely identifies a tuple in a relation.

Note :

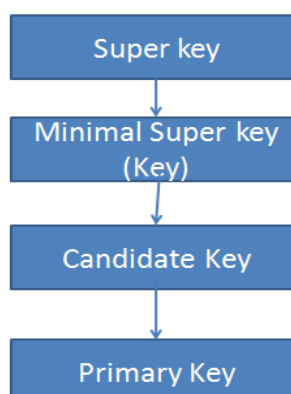
A Relation(table) can have many Superkeys, and also many Minimal Superkeys.

If a Relation(table) has more than one Minimal Superkeys each can be called as Candidate Keys.

One of the candidate keys is arbitrarily designated to be the primary key, and the others are called secondary keys(or Alternative key).

Key Hierarchy

- Primary key doesn't allow duplicates and Null Values.



Foreign Key:

A foreign key is an attribute whose values match the primary key values in the related table.

Foreign Key (Referential Integrity Constraint) :

Referential Integrity Constraint is used to specify interdependencies between relations. This constraint specifies a column or list of columns as a foreign key of the referencing table.

- A **foreign key** means the values of a column in one table must also appear in a column in another table. The foreign key in the child table will generally reference a **primary key** in the parent table. The referencing table is called the child table & referenced table is called the parent table
- Self Referential Integrity Constraint mean a column in one table references the primary key column(s) in the same table.

EMPReferencing relation

Eno	Ename	Salary	Dept_no (FK)	Voter_Id
1	raju	20000	10	V12345
2	ravi	40000	10	V12222
3	Raju	25000	20	V45666

In EMP table Eno is Primary Key. (Duplicates and Null values are not allowed in Eno)

In EMP table Dept_no is foreign key which references DEPT table Dept_no column. (A value for Dept_no in EMP table accepts only if it exists in Dept_no column in DEPT table.)

DEPT

Referenced relation

Dept_no (PK)	Dname	Dloc
10	MTech	BVRM
20	MBA	HYD
30	MCA	BVRM

In DEPT table Dept_no is Primary key.

Secondary Key:

The secondary key is defined as a key that is used to for data retrieval purpose.

Example: In the customer table the data retrieval can be facilitated when CUST_LAST and CUST_PHONE number are used.

Integrity Rules:

Integrity rules are used in the database design.

1. Entity Integrity: All primary key entries are unique and no part of the primary key may be NULL.
Example: In Agent table the agent_code is primary key and this column is free from null values.
2. Referential Integrity: A foreign key is an attribute whose values match the primary key values in the related table.
Example: The vendor_code is the primary key in the vendor key and it occurs as a foreign key.
3. NOT NULL: NOT NULL constraint can be placed in a column while inserting a row that column must have a value.
4. Unique: Unique constraint can be placed in a column while inserting a row that column have unique values. (No duplication).

RELATIONAL SET OPERATORS:

The relational set operators are SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, DIVIDE.

1. UNION:

The UNION operator combines all rows from two tables, excluding duplicate rows. The tables must have the same structure.

Product 1

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

Product 2

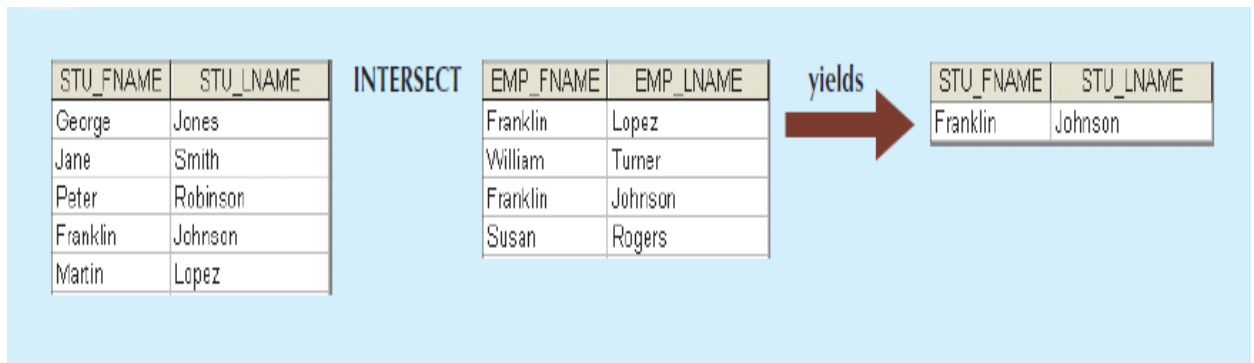
P_CODE	P_DESCRIPT	PRICE
345678	Microwave	160.00
345679	Dishwasher	500.00

Query: Product 1 union product 2

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99
345678	Microwave	160
345679	Dishwasher	500

2. INTERSECT:

The INTERSECT operator gives only the rows that appear in both tables. The tables are also have same structure.



3. DIFFERENCE:

The DIFFERENCE operator gives all rows in one table that are not found in other table.

STUDENT

EMPLOYEE

STU_FNAME	STU_LNAME	EMP_FNAME	EMP_LNAME
George	Jones	Franklin	Lopez
Jane	Smith	William	Turner
Peter	Robinson	Franklin	Johnson
Franklin	Johnson	Susan	Rogers
Martin	Lopez		

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

QUERY: STUDENT MINUS EMPLOYEE

4. PRODUCT:

The PRODUCT operator gives all possible pair of rows from two tables.

P_CODE	P_DESCRIPTION	PRICE	PRODUCT				P_CODE	P_DESCRIPTION	PRICE	STORE	aisle	shelf
123456	Flashlight	5.26				yields →	123456	Flashlight	5.26	23	W	5
123457	Lamp	25.15	23	W	5		123456	Flashlight	5.26	24	K	9
123458	Box Fan	10.99	24	K	9		123456	Flashlight	5.26	25	Z	6
213345	9v battery	1.92	25	Z	6		123457	Lamp	25.15	23	W	5
254467	100W bulb	1.47					123457	Lamp	25.15	24	K	9
311452	Powerdrill	34.99					123457	Lamp	25.15	25	Z	6
							123458	Box Fan	10.99	23	W	5
							123458	Box Fan	10.99	24	K	9
							123458	Box Fan	10.99	25	Z	6
							213345	9v battery	1.92	23	W	5
							213345	9v battery	1.92	24	K	9
							213345	9v battery	1.92	25	Z	6
							311452	Powerdrill	34.99	23	W	5
							311452	Powerdrill	34.99	24	K	9
							311452	Powerdrill	34.99	25	Z	6
							254467	100W bulb	1.47	23	W	5
							254467	100W bulb	1.47	24	K	9
							254467	100W bulb	1.47	25	Z	6

5. SELECT:

The SELECT operator gives all rows that satisfies a given condition.

P_CODE	P_DESCRIPTION	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

SQL> Select product where P_CODE=311452.

P_CODE	P_DESCRIPTION	PRICE
311452	Powerdrill	34.99

6. PROJECT:

The PROJECT operator gives all values for selected attributes. In other words project gives a vertical subset of tables.

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

Project P_DESCRIPT Product;

P_DESCRIPT
Flashlight
Lamp
Box Fan
9v battery
100W bulb
Powerdrill

7. JOIN: The JOIN operator combines rows from 2 or more tables. There are several types of joins.

a. Natural Join

A Natural Join joins tables by selecting the rows with common values in their common attributes. A natural join is the result of a three-stage process:

1). First, a PRODUCT of the tables is created, yielding the results shown in Figure 3.12.

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Query: SQL> Product NATURAL JOIN Vendor

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1132445	Walker	32145	231	125	6152439887
1132445	Walker	32145	231	167	6153426778
1132445	Walker	32145	231	231	6152431124
1132445	Walker	32145	231	333	9041234445
1217782	Adares	32145	125	125	6152439887
1217782	Adares	32145	125	167	6153426778
1217782	Adares	32145	125	231	6152431124
1217782	Adares	32145	125	333	9041234445
1312243	Rakowski	34129	167	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1312243	Rakowski	34129	167	231	6152431124
1312243	Rakowski	34129	167	333	9041234445
1321242	Rodriguez	37134	125	125	6152439887
1321242	Rodriguez	37134	125	167	6153426778
1321242	Rodriguez	37134	125	231	6152431124
1321242	Rodriguez	37134	125	333	9041234445
1542311	Smithson	37134	421	125	6152439887
1542311	Smithson	37134	421	167	6153426778
1542311	Smithson	37134	421	231	6152431124
1542311	Smithson	37134	421	333	9041234445
1657399	Vanloo	32145	231	125	6152439887
1657399	Vanloo	32145	231	167	6153426778
1657399	Vanloo	32145	231	231	6152431124
1657399	Vanloo	32145	231	333	9041234445

2). Second, a SELECT is performed on the output of Step a to yield only the rows for which the AGENT_CODE values are equal. The common columns are referred to as the **join columns**.

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124

3). A PROJECT is performed on the results of Step b to yield a single copy of each attribute, thereby eliminating duplicate columns.

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

The final outcome of a natural join yields a table that does not include unmatched pairs and provides only the copies of the matches.

b. Equijoin:

In Equijoin the tables on the basis of equality condition that compares specified columns of each table. In Equijoin the comparison operator Is Equal To is used in the condition.

Or

Inner join produces only the set of records that match in both Table A and Table B.

Product:

Prod_code	Prod_Descp	Vend_code
123	tyres	V101
124	tubes	V102
125	Bolts	-----

Vendor:

Vend_code	Vend-name
V101	ravi
V102	ram
V103	krishna

Result:

Prod_code	Prod_Descp	Vend_code	Vend-name
123	Tyres	V101	ravi
124	Tubes	V102	ram

Outer Join:

In the Outer Join the matched pair of records would be written and any unmatched values in other table would be NULL.

c. Left Outer Join:

The Left Outer Join matched the records would be return and any unmatched values in the other table would be NULL.

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124
1542311	Smithson	37134	421	

d. Right Outer Join:

In Right Outer Join the matched records would be and any unmatched values in the right table would be NULL.

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124
			333	9041234445

e. Full Outer Join:

Full outer join produces the set of all records in Table A and Table B, with matching records from both sides where available. If there is no match, the missing side will contain null.

Product:

Prod_code	Prod_Descp	Vend_code
123	tyres	V101
124	tubes	V102
125	Bolts	-----

Vendor:

Vend_code	Vend-name
V101	ravi
V102	ram
V103	krishna

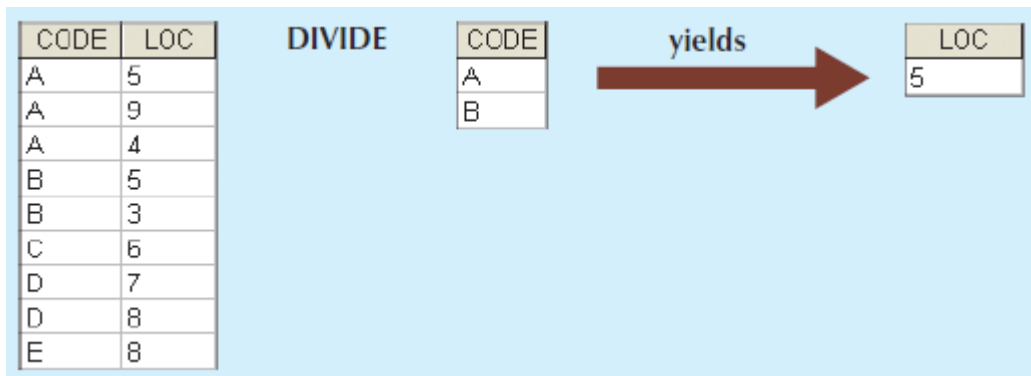
Result:

Prod_code	Prod_Descp	Vend_code	Vend-name
123	Tyres	V101	ravi

124	Tubes	V102	ram
125	Bolts	-----	-----
-----	-----	V103	krishna

8. DIVIDE:

The DIVIDE operator uses one single column table as a divider and two column table as the dividend. The output of DIVIDE operator is a single column with a values column-A from the dividend table rows where the values of the common column in both tables match.



Data Dictionary:

The Data Dictionary provides a description of all tables in the database. The Data Dictionary contains attribute names and characteristics of each table in the system. The data dictionary contains meta data.

Example:

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENCED TABLE
CUSTOMER	CUS_CODE	Customer account code	CHAR(5)	99999	10000-99999	Y	PK	
	CUS_LNAME	Customer last name	VARCHAR(20)	Xxxxxxx		Y		
	CUS_FNAME	Customer first name	VARCHAR(20)	Xxxxxxx		Y		
	CUS_INITIAL	Customer initial	CHAR(1)	X				
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	dd-mmm-yyyy				
	AGENT_CODE	Agent code	CHAR(3)	999				

System Catalog:

Like the Data Dictionary, System Catalog contains metadata. The system catalog describes table names, table creator, and creation of data, Number of column in each table, the data types of the column, authorized users and access privileges.

Relationships within Relational Databases

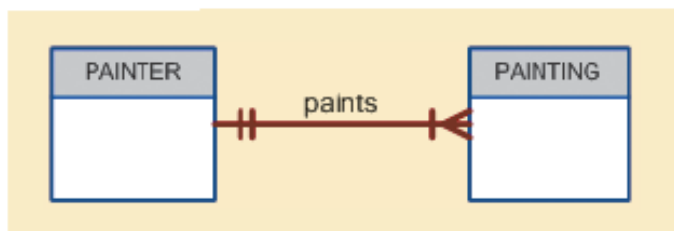
The relationships are classified into One-to-Many, One-to-One and Many-to-Many.

One-to-Many (1:M) relationship

One entity of one table is associated with number of entities into other tables.

Consider the "Painter Paints Paintings" the ER Model and implementations are shown below.

The 1:M relationship between PAINTER and PAINTING



The implemented 1:M relationship between PAINTER and PAINTING

Table name: PAINTER

Primary key: PAINTER_NUM

Foreign key: none

PAINTER_NUM	PAINTER_LNAME	PANTER_FNAME	PANTER_INITIAL
123	Ross	Georgette	P
126	Itero	Julio	G

Table name: PAINTING

Primary key: PAINTING_NUM

Foreign key: PAINTER_NUM

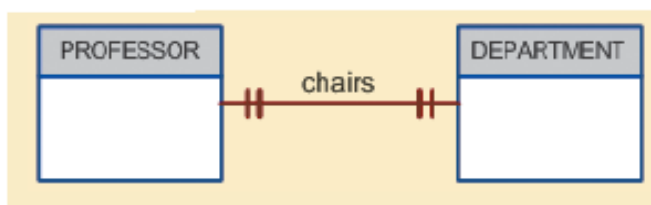
PAINTING_NUM	PAINTING_TITLE	PAINTER_NUM
1338	Dawn Thunder	123
1339	Vanilla Roses To Nowhere	123
1340	Tired Flounders	126
1341	Hasty Exit	123
1342	Plastic Paradise	126

One-to-One

In this relationship one entity of one table is associated with one entity of other table and vice-versa.

Consider the PROFESSOR AND DEPARTMENT. The ER data model and implementation are shown in below.

The 1:1 relationship between PROFESSOR and DEPARTMENT

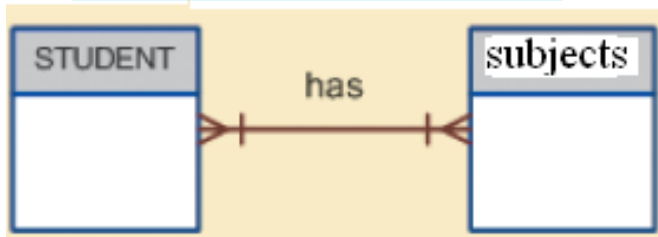


May-to-Many (M:M) relationship

In this relationship each and every entity of one file will be associated with one or more entities of another tables and vice versa. This relationship is not supported in relational environment. The many to many relationship can be converted into two One to Many relationships.

Consider student and subjects example. The ER model and implementation are shown in below.

The ERM's M:N relationship between STUDENT and CLASS



In above each and every entity of student file is associated with one or more entities of the subject table because each student will opt one or more subjects in a semester.

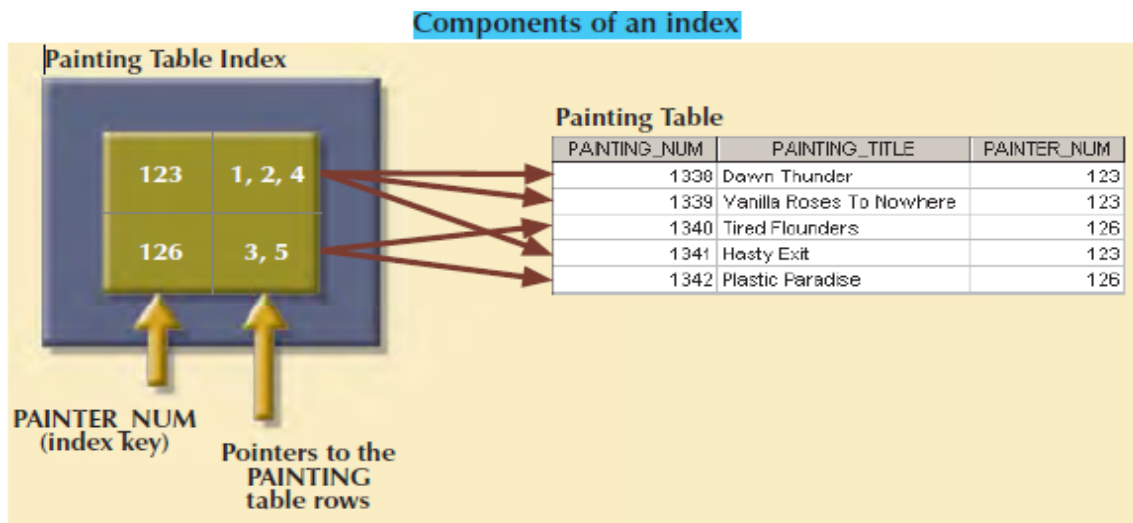
Each and every entity of subject file is associated with one or more entities of student table because each sub will be opted by more than one student in a semester. Many to Many associations are not supported.

Hence the relationship will be converted into two one to many associations as shown below by introducing an intermediate table in which the common data of the original file are stored.

Index:

An Index is composed of an index key and set of pointers. Each key points to the location of data identified by the key.

Example: Suppose we want to look up all of the paintings created by the given painter without an index, we must read each row in the painting table. If we index the painter table and use the index key of painter number, we look up appropriate painter number in the index and find the matching pointers.



DBMSs use indexes for many different purposes. You just learned that an index can be used to retrieve data more efficiently. But indexes can also be used by a DBMS to retrieve data ordered by a specific attribute or attributes.

For example, creating an index on a customer's last name will allow you to retrieve the customer last name in alphabetical order.

Indexes play an important role in DBMS for implementation of primary keys. When we define a table primary key the DBMS automatically creates an unique index on the primary key column.

When we declare the customer-code to be the primary key of the customer table, the DBMS automatically creates a unique index on that attribute. An unique index is an index in which the index key can have one pointer.

CODD`S RELATIONAL DATABASE RULES

Dr E F Codd published a list of 12 rules to define a relational database.

Rule 1: Information:

All information in a relational database must be logically represented column values in rows with tables.

Rule 2: Guaranteed Access:

Every value in a table is guaranteed to be accessible through a combination of table name, primary key and column name.

Rule 3: Systematic Treatment of NULLs:

NULL must be represented and treated in a systematic way (Independent of data type).

Rule 4: Dynamic online catalog based on the relation:

The metadata must be stored as ordinary data in a table within the database. Such data must be available to authorized users.

Rule 5: Comprehensive data sub language:

The relational database may support many languages. However it must support data definition, view definition, data manipulation integrity constraints, authorizations and transaction management.

Rule 6: View Update:

Any view i.e., theoretically must be updatable through the system.

Rule 7: High level Insert, Update and Delete:

The database must support insert, update and delete.

Rule 8: Physical data independency:

Application programs are logically unaffected when storage structures are changed.

Rule 9: Logical data independency:

Application programs are logically unaffected when changes are made to the table structures.

Rule 10: Integrity Independency:

All relational integrity constraints must be definable in the relational language and stored in the system catalogs.

Rule 11: Distribution Independency:

The end users and application programs are unaffected by the data locations.

Rule 12: Non Sub Version:

If the system support low level access to the data, there must not be a way to bypass the integrity rules of the database.

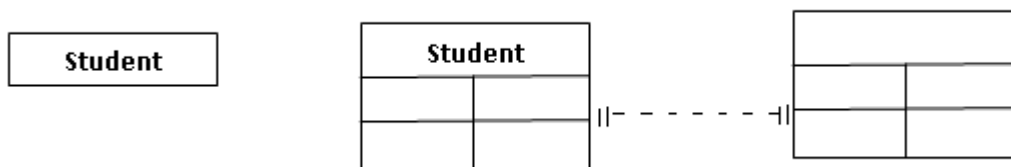
Entity Relationship Modeling

The Entity Relationship modeling forms an Entity Relationship diagram. The ERD represents the conceptual database. The Entity Relationship diagram contains mainly three components. Entity, Attribute and Relationship.

Entity: An Entity represents a real world object.

Eg: Student, Customer, Employee, Subject, Faculty, and Product

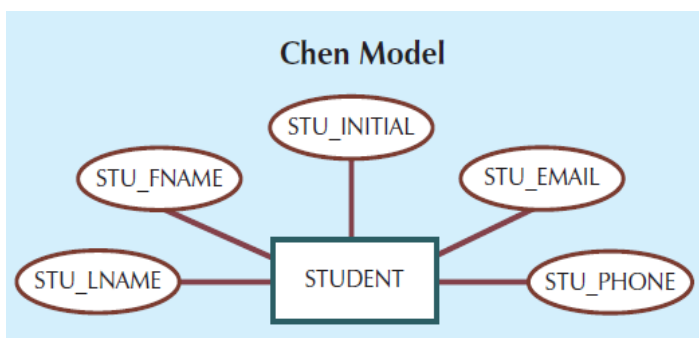
In chen and crow foot notation an entity is represented by rectangle containing entity name. The entity names usually return in capital letters.



Attributes: An Attribute represents characteristics are properties of an entities.

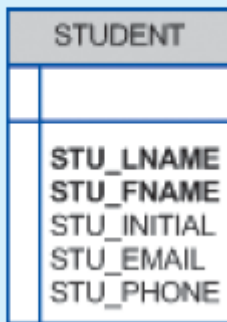
Eg: stu_name , stu_address

In chen notation the attributes are represented by Vowels and are connected to the entity rectangle with a line .Each vowel contains name of the Attribute.



In the cross foot Notation the attributes are written in the attribute box below the entity rectangle.

Crow's Foot Model

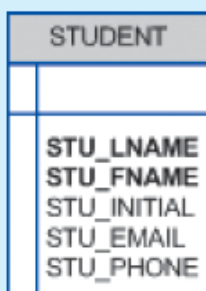


Required and Optional Attribute:

A Required is an Attribute that must have a value. An Optional Attribute is an Attribute that does not have a value.

In the Crows foot Notation the required Attributes are represented by bold face.

Crow's Foot Model



Domain: Attributes have a domain. A domain is the set of possible values for a given Attributes.

Eg: The domain for the gender attribute consists of only two possibilities namely male and female.

Identifiers: Identifiers contains one attributes that uniquely identify an entity in the entity set. In relational model, such identifiers are mapped to primary keys in the table. Identifiers are underlined in the ER diagram.

For Eg. STUDENT(Sto_no, Stu_fname, Stu_Lname, Stu_email);

Composite Identifier: A Composite Identifier is a primary Key contains more than one attribute.

For Eg. To identify the each student entity by using a composit primary key composed of the combination of Stu_Lname and Stu_Fname instead of using Stu_no.

Composite and Single Attribute: Attributes are classified as single Attribute or Composite Attribute.

A Single Attribute is an attribute that cannot be sub divided.

Eg: Age, F_name , Gender.

A Composite Attribute is an Attribute that can be further sub divided which gives additional Attributes.

Eg: Address can be sub divided into street , city and pincode.

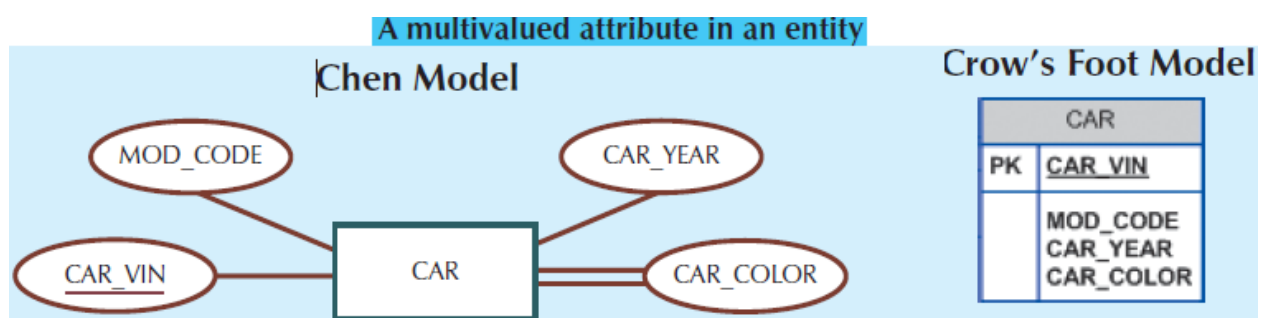
Single Attribute: A Single Attribute ia an attribute that can have only one value.

Eg: STU_Number in the STUDENT TABLE.

Multi Value Attribute: A multi value Attribute is an attribute that can have many values.

Eg: A car color may be sub divided into many colors.

In Chenn notation , the multivalued attributes are shown by a double line connecting to the attribute to entity.

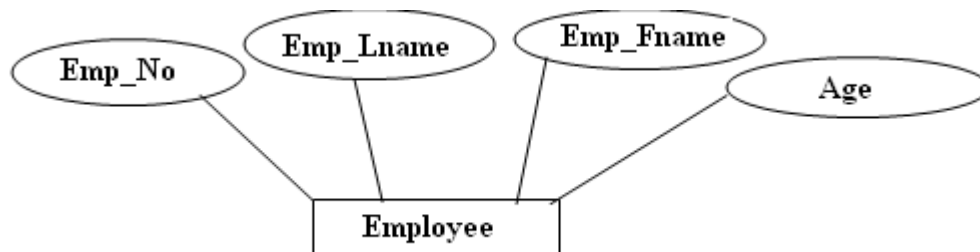


Note: The crows foot notation does not identify multivalued attributes.

Derived Attribute: A Derived Attribute is an attribute whose values are calculated from other Attributes.

Eg: An Employee age computing difference between current data and the employee date of birth.

A Derived Attribute is indicated in a Chen's Notation by a dashed line by the connecting to the attribute and entity.



Relationship: The Relationship is an Association between entities.

Eg: A professor teaches a class.

There are three different types of relationships

1. **One- One**
2. **One- Many**
3. **Many-Many**

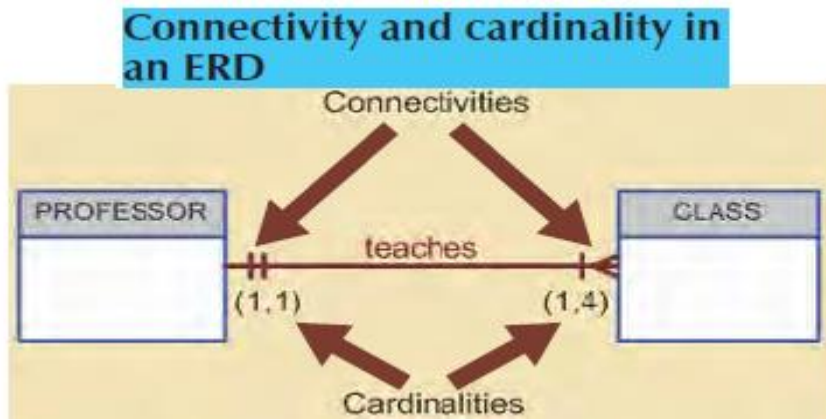
CONNECTIVITY AND CARDINALITY:

Cardinality Express the minimum and maximum number of entities occurrences associated with one occurrence of the related entity.

In the ERD the cardinality is indicated by placing the appropriate numbers beside the entities using the format(x, y).

The first value represents the minimum number of associated entities. The second value represents the maximum number of associated entities.

For Eg:



The cardinality (1,1) indicates that each class is taught by only one professor.

The cardinality (1,4) indicates one entity in the professor relation associate not more than four times in the entities of the class Relation.

Existence Dependency: An Entity is said to be existence dependency when the entity is associated with another related entity.

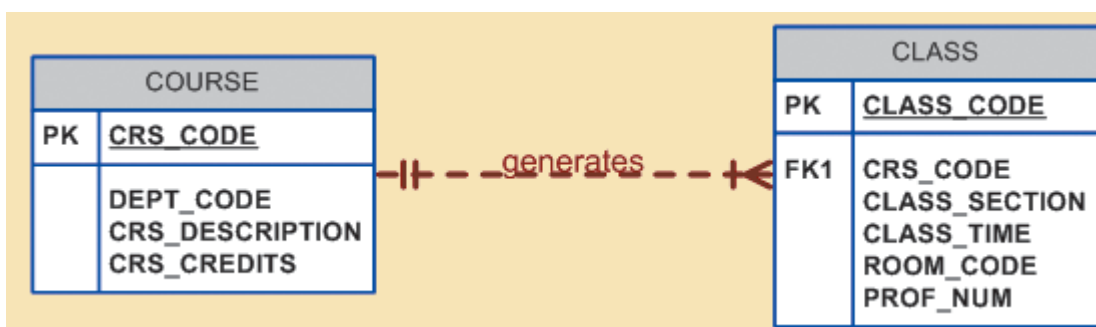
Weak Relationship

A weak relationship exists if the primary key of the entity does not contain component of the parent entity.

COURSE(CRS_CODE, DEPT_CODE, CRS_DESCRIPTION, CRS_CREDIT)

CLASS(CLASS_CODE, CRS_CODE, CLASS_SECTION, CLASS_TIME, ROOM_CODE, PROF_NUM)

The Relationship between course and class is weak because the class_code is the primary key in the class entity while the course_code in the class is foreign key. In this example the class primary key did not inherit the primary key components from the course entity.



STRONG RELATIONSHIP

A Relationship exist if the primary key of the related by contains a primary key component of the parent entity. The relationship between course and class is strong because the class entity composite entity key is composed of class_code+ Crs_code. The class primary key inherit the primary key component from the course entity. In the strong relationship we can write 'o' symbol next to the entity.

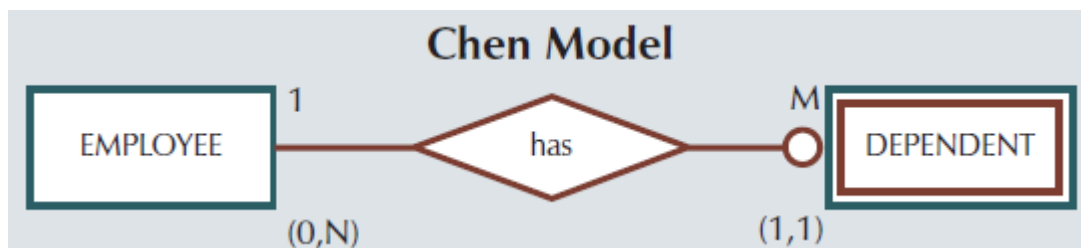
WEAK ENTITY

In Weak Entity the primary key is primary key is partially as totally derived from the parent entity in the relationship .For example: The dependent entity key was inherited from the employee entity as show below.

Employee (Emp_no, Emp_LName, Emp_FName,Emp_init,Emp_dob).

Dependent (Emp_no, Dept_no, Dept_LName,Dept_FName, Dept_dob).

In chen Notation the Weak entity is represented by double weak entity.



STRONG ENTITY

In Strong entity the primary key has not partially or totally derived from the parent entity in the relationship. For example in the course and class relationship the class table primary key is class code which is not derived from the course parent key. The entity class is a strong entity.

Class(class_code, course_code,class_desc,class_time, prof_code)

In the Course and Class relationship is composed of class_code and course_code is derived from the course parent entity. The entity class is a weak entity.

Class(class_code, course_code, desc,class_time, prof_code)

RELATIONSHIP PARTICIPATION:

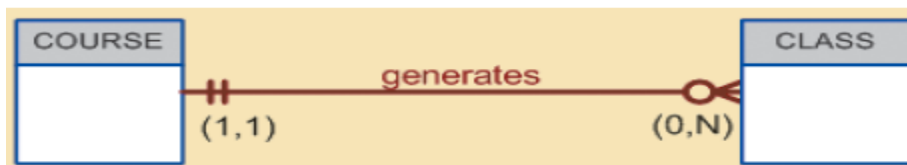
The Relationship Participation between the entities is either optional or mandatory.

Optional participation

Optional participation means that one entity occurrence does not require a corresponding entity occurrence in a particular Relationship.

Eg: "course generate class" relationship an entity occurrence in the course table does not necessary require the corresponding entity occurrences in the class table.

In the crows foot notation an optional relationship between entities is shown by drawing a circle on the side of optional entity. The minimum cardinality is zero for the optional entity.

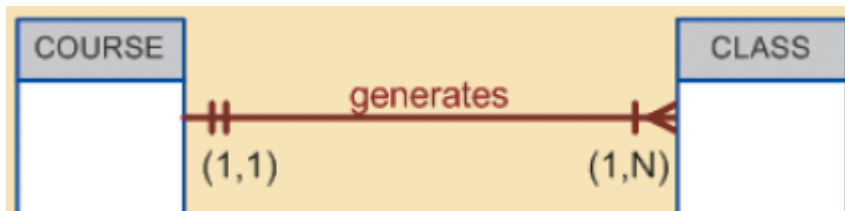


HEAR CLASS IS OPTIONAL TO COURSE

Mandatory participation means that one entity occurrence requires a corresponding entity occurrence in a particular relationship.

Eg 1: "course generate class" relationship an entity occurrences in the course table necessary require the corresponding entity occurrences in the class table.

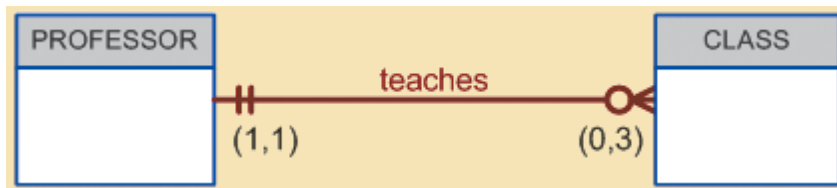
In the crows foot Notation there is no separate symbol for the mandatory entity. The minimum cardinality is one for the mandatory entity.



(HEAR COURSE AND CLASS IS MANDATORY RELEATIONSHIP)

Eg 2: The "Professor teaches class" relationship it possible for a professor not to teach a class. Therefore class is optimal to the professor on other hand a class a class must

be taught by a professor. Therefore the professor is mandatory on other hand a class must be taught by a professor. Therefore the professor is mandatory a class.



The cardinality (1,1) represents one class is taken by one professor. The cardinality (0,3) indicates the professor may teach no classes or theory classes.

The following table shows various cardinalities that are supported by crow's foot notation.

Crow's Foot Symbols		
CROW'S FOOT SYMBOL	CARDINALITY	COMMENT
	(0,N)	Zero or many. Many side is optional.
	(1,N)	One or many. Many side is mandatory.
	(1,1)	One and only one. 1 side is mandatory.
	(0,1)	Zero or one. 1 side is optional.

RELATIONSHIP DEGREE

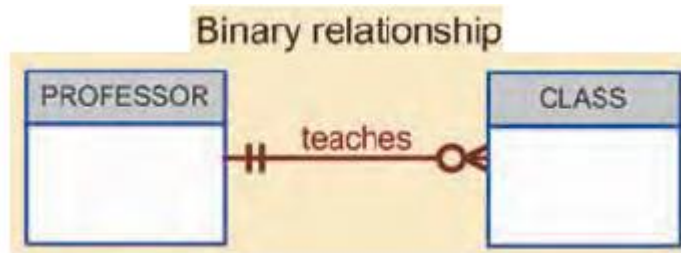
A Relationship Degree indicates the number of entities associated with a relationship.

A Unary Relationship: A Unary Relationship exists when an association is maintained within a single entity.



Eg: An Employee within the employee entity is the manger for one or more entities within that entity.

Binary Relationship: A binary relationship exists when two entities are associated.



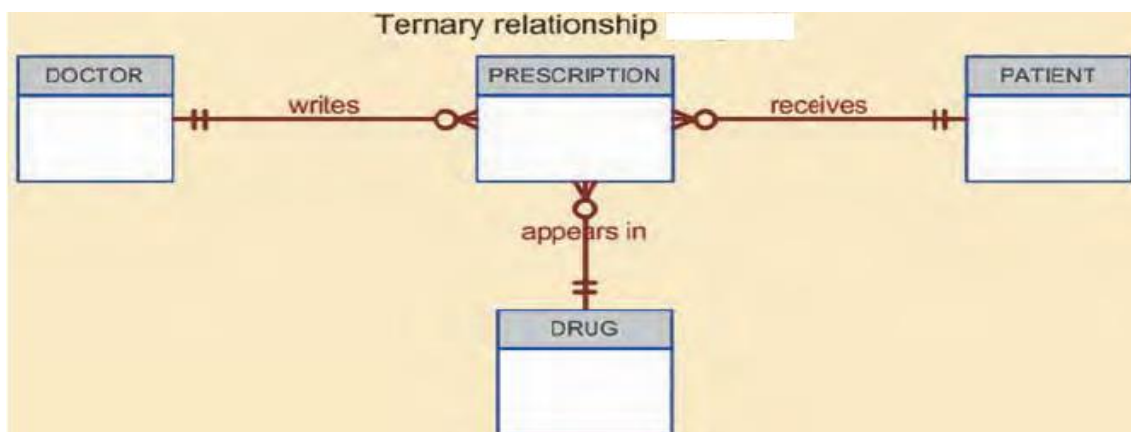
Eg: The relationship a professor teaches one or more class.

Ternary Relationship: A ternary relationship exists when three entities are associated.

Eg: A doctor writes one or more prescription

A patient may receive one or more prescription

A drug may appear one or more prescription



Recursive Relationship: A Recursive Relationship is one in which a relationship between the same entity set.

There are three types of Recursive Relationships

1. **One- One**
2. **One- Many**
3. **Many-Many**

One to One: A One to One unary relationship may be expressed by an employee may be married to one and only one other employee.



One to Many: A One to Many unary relationships may be expressed by an employee may manage many employees.

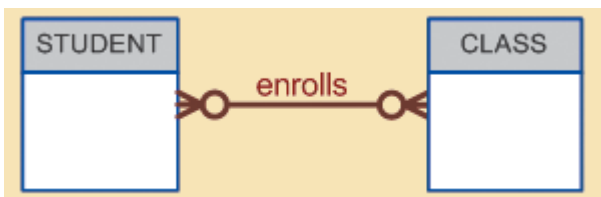


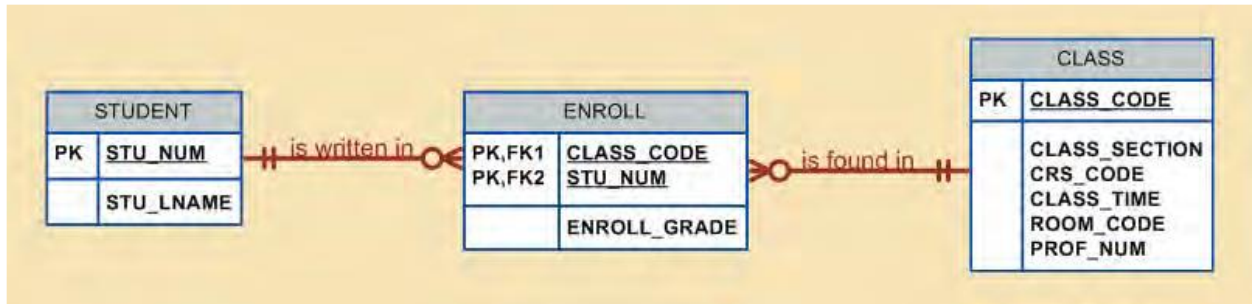
Many to Many: The Many to Many relationship may be expressed by a course may be pre requisite to many other courses.



Associative Entities: The associative Entity is used to implement a many to many relationship between entities. This associated entity is composed of the primary key of each of the entites to be connected.

Example: The Crown foot notation the relationship between the parent and child entities indicates the strong relationship.





DEVELOPING AN ER DIAGRAM:

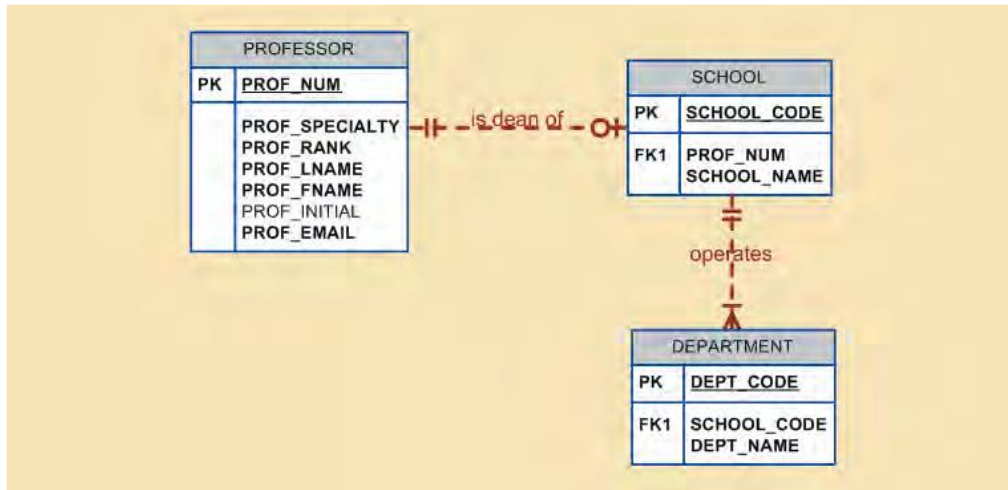
The process of database design is an Iterative process rather than a sequential process. Building an ER diagram, usually involves the following Activities.

1. Create a detailed description of operation of the organization.
2. Identify the business rules from the description.
3. Identify entities, Attributes and relationship from the business rules.
4. Developing the initial ERD.
5. Identify the attributes and primary keys.
6. Revise and review the ERD

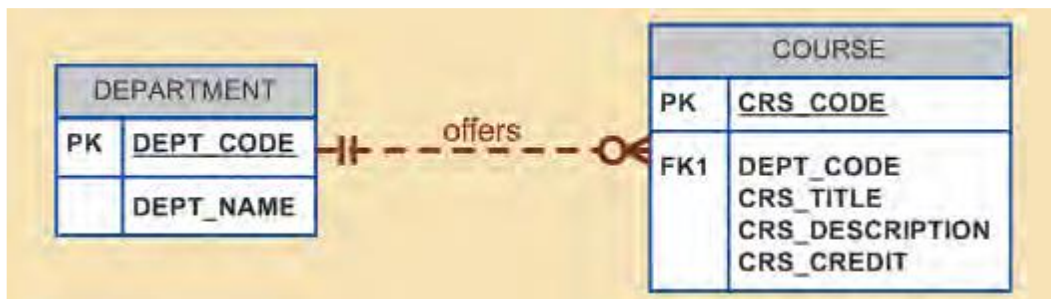
During the review process additional objects, attributes and relationships will be covered. Therefore the basic ERD will be modified to incorporate the newly discovered components.

Eg: Let us with initial interviews with the tiny college administrator and the interview process gives the following business rules.

1. Tiny college is divided into several schools a school of business, a school of Arts and Science, a school of education a school of applied sciences, each school is administrated by a deal who is a professor.
2. Each school is composed of several departments.



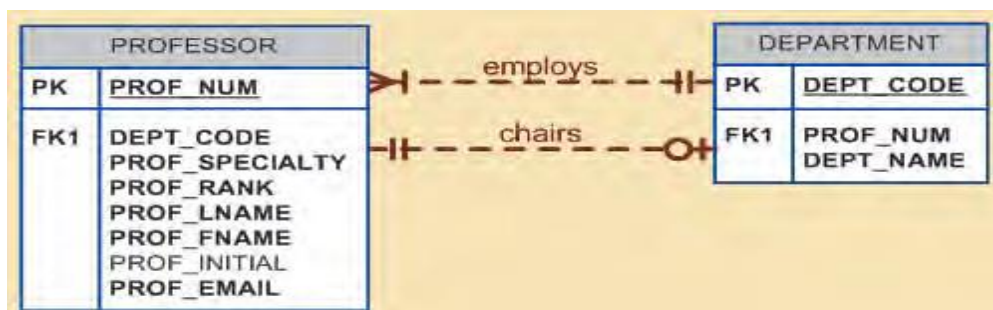
3. Each department may offer several courses.



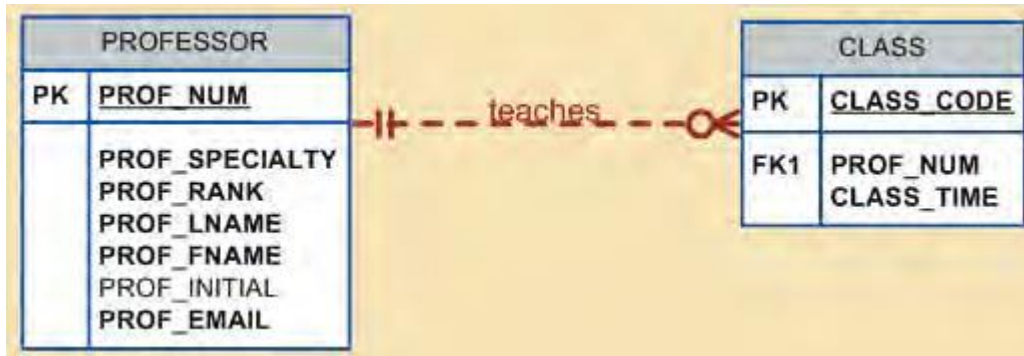
4. Each course may operate several classes.



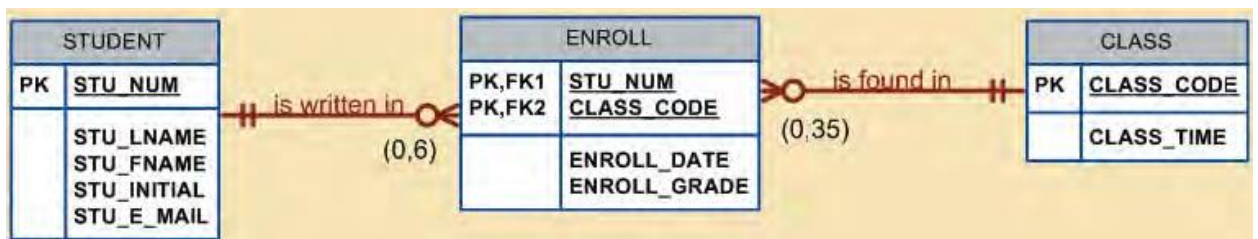
5. Each department may have professors one and only one of those professors chairs the department and no professor is required to accept the chair position.



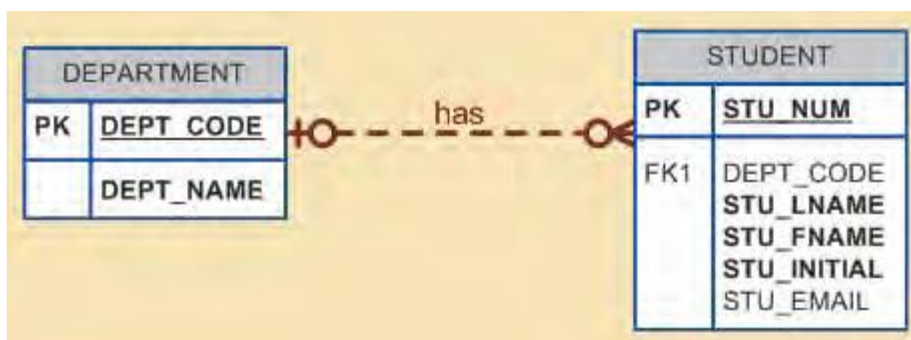
6. Each professor may teach the classes. A professor may not teach the class.



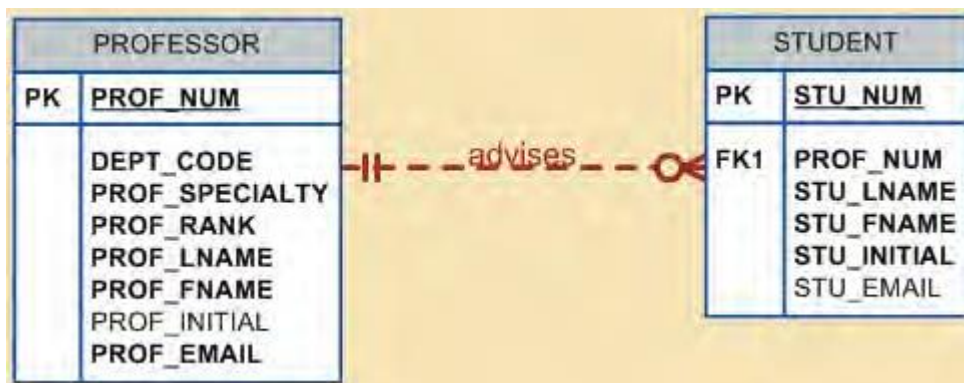
7. A student may enroll several classes, each class contains several students. Student is optional to class in the many to many relationships. This many to many relationship must be divided into two one to many relationship through many enroll entities.



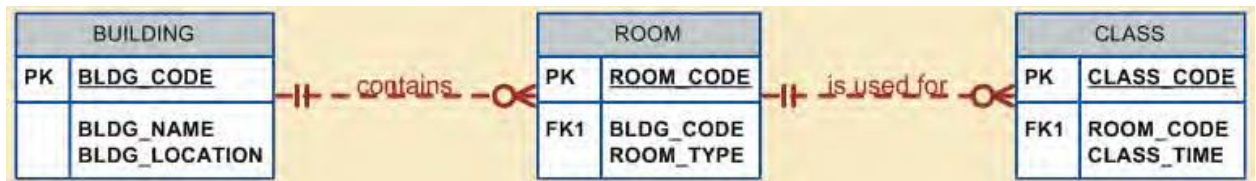
8. Each department has several students.



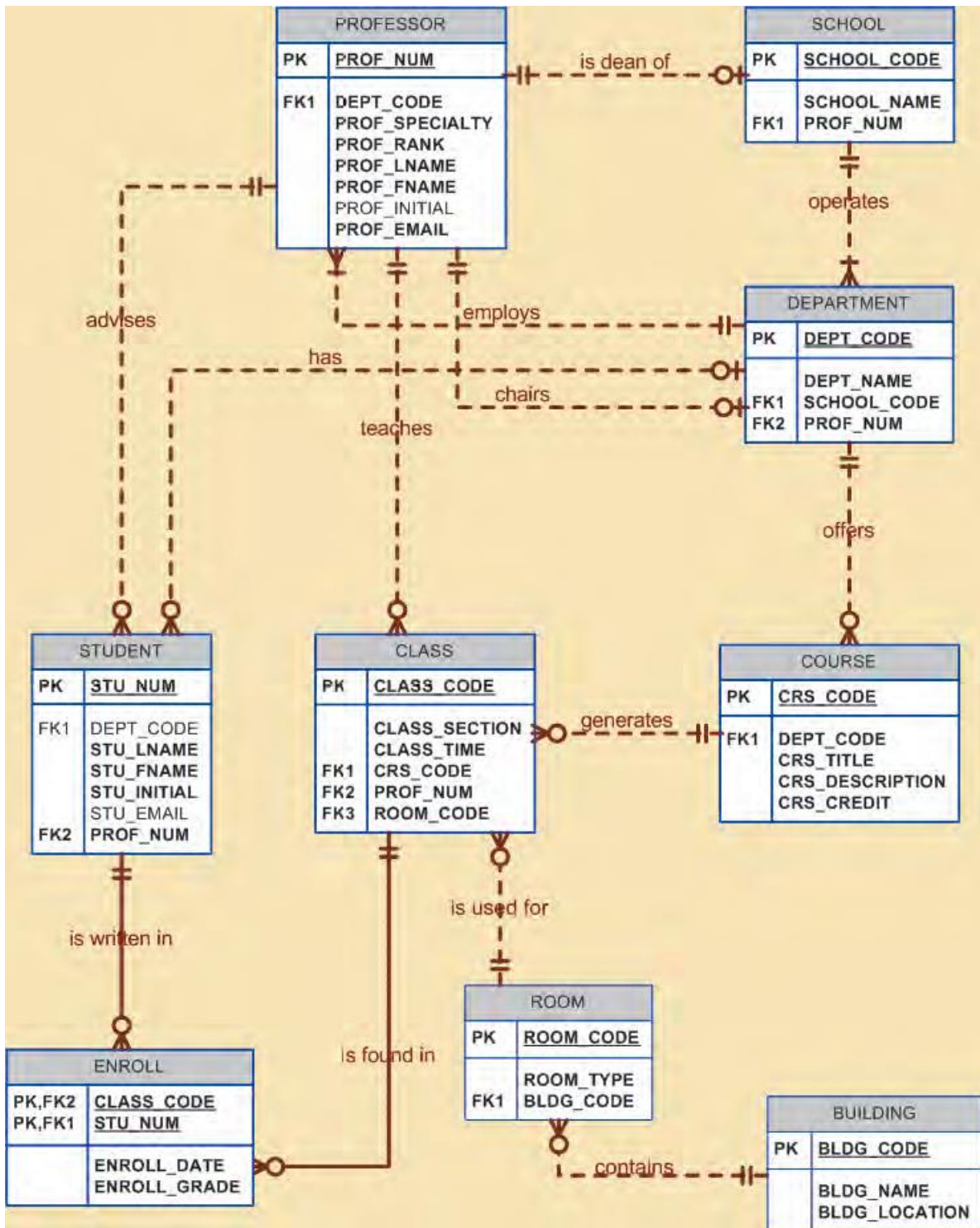
9. Each student has an Adviser in Department, each Adviser consists several students. An Adviser is also a professor, but not all professor advice students.



10. A building contains Rooms and room is used for classes.



The following diagram shows crows foot ERD for tiny college.



DATABASE DESIGN CHALLENGES: CONFLICTING GOALS

Database Design Challenges:

Database designers often must make design compromises that are triggered by conflicting goals, such as adherence to design standards (design elegance), processing speed, and information requirements.

Design standards:

The database design must conform to design standards. Such standards have guided you in developing logical structures that minimize data redundancies.

In short, design standards allow you to work with well-defined components and to evaluate the interaction of those components with some precision.

Processing speed:

In many organizations, particularly those generating large numbers of transactions, high processing speeds are often a top priority in database design. High processing speed means minimal access time. If the focus is on data-retrieval speed, you might also be forced to include derived attributes in the design.

Information requirements:

The quest for timely information might be the focus of database design.

Complex information requirements may dictate data transformations, and they may expand the number of entities and attributes within the design.

Therefore, the database may have to sacrifice some of its "clean" design structures and/or some of its high transaction speed to ensure maximum information generation.

A design that meets all logical requirements and design conventions is an important goal.

However, if this perfect design fails to meet the customer's transaction speed and/or information requirements, the designer will not have done a proper job from the end user's point of view.

Compromises are a fact of life in the real world of database design. Even while focusing on the entities, attributes, relationships, and constraints, the designer should begin thinking about end-user requirements such as performance, security, shared access, and data integrity.

Finally, prepare the document! Put all design activities in writing. Then review what you've written.

Normalization of Database Tables

Database Tables & Normalization:

1. In Database designed process, the table is the basic building block.
2. The ER model gives good table structure. But it is possible to create poor table structure. Even in a good database structure design.

Def:

Normalization is an Analysis of functional dependency between the attributes of a relation. It reduces the complex user view into set of stable sub groups or fields.

The normalization process is used to create a good table structure to minimize data redundancy.

Normalization works through a series of stages called normal form.

The first three stages are

- First Normal Form(1NF)
- Second Normal Form(2NF)
- Third Normal Form(3NF)

Business Databases: Business databases are sufficient to normalize to 2NF or 3NF. The other stages are

- Boyce Code Normal Form (BCNF)
- Fourth Normal Form(4NF)
- Fifth Normal Form (5NF)

Normalization is a very important in database design .Generally the higher normal forms, the more relational join operations required to produce a specific output. Therefore occasionally we accepted to denormalize some positions of the database to increase the efficiency.

DeNormalization produces a lower normal form i.e., 3NF will be connected into 2NF will be converted into 1NF.

The need for Normalization:

Consider the Database activities of a construction company that manages several building projects. Each Project has its own project number and project name, employee assigned to it and soon. Each employee has employee number, employee name, classification etc.

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS	Total_Charge
15	Evergreen	103	June E .Arbough	Elect.Engineer	84.50	23.8	2011.1
		101	Swetha	Database designer	105.00	19.4	2037
		105	Lakshmi	Database designer	105.00	35.7	3748.5
		106	Durga	Programmer	35.75	12.6	450.45
		102	Ram	Systems Analyst	96.75	20.8	2012.4
SubTotal 10259.45							
18	Amber wave	114	Harika	Applications designer	48.10	25.6	1231.36
		118	Ganesh	General support	18.36	45.3	831.708
		104	Sri	Systems analyst	96.75	32.4	3134.7
		112	Hari	DSS Analyst	45.95	44.0	2021.8
Sub Total							7219.568
22	Rolling Tide	105	Sruthi	Database designer	105.00	64.7	6793.5
		104	Raju	Systems analyst	26.75	48.4	1294.7
		113	Ravi	Application designer	48.10	23.6	1135.16
		111	Ramesh	Clerical support	26.87	22.0	591.14
		106	Rao	Programmer	35.75	12.8	457.6
Sub Total							10272.1
25	Starflight	107	Rekha	Programmer	35.75	24.6	879.45
		115	Rani	System analyst	96.75	45.8	4431.15
		101	John	Database designer	105.00	56.3	5911.5
		114	Manikanta	Applications designer	48.10	33.1	1592.11
		108	Nalini	System analyst	96.75	23.6	2283.3
		118	James	General secretary	18.36	30.5	559.98
		112	P J	DSS Analyst	45.95	41.4	1902.33
Sub Total							17559.82
Total Amount							45310.94

The Easiest way to generate the required report to create a table that table has some fields of the Report.

Table_Name : Construction_Company

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
		101	John G. News	Database Designer	105.00	19.4
		105	Alice K. Johnson *	Database Designer	105.00	35.7
		106	William Smithfield	Programmer	35.75	12.6
		102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
		118	James J. Frommer	General Support	18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
		111	Geoff B. Wabash	Clerical Support	26.87	22.0
		106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
		115	Travis B. Bawangi	Systems Analyst	96.75	45.8
		101	John G. News *	Database Designer	105.00	56.3
		114	Annelise Jones	Applications Designer	48.10	33.1
		108	Ralph B. Washington	Systems Analyst	96.75	23.6
		118	James J. Frommer	General Support	18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	45.95	41.4

An Employee can be assigned more than one project.

For example: Employee number 104 has been assigned to two project .Therefore knowing the project _no and employee no will find the job classification and hours worked. Therefore project_No and emp_no will be taken as primary key.

The above structure of the table has the following deficiency

1. The project _no is a part of primary key. But it contains null values.
2. The table entries invites data inconsistency for example job classification value Electrical_Engineer might be entered.Elec_engi ,EE
3. The table displays data redundancy.

Update Anomalies: Modify the job class for Employee_No 105 requires many alternatives.

Insertion Anomalies: To complete a row definition of a new employee must be assigned to a project. If the employee is not assigned, a dummy project must be created to complete the row.

Deletion Anomalies: Suppose only one employee is associated with a project, if that employee leaves the company and the employee data are deleted, the project information will also be deleted.

The above deficiency of table structure appears to work, the report gives different results depending on data.

Normalization Process: The most common Normal forms and their characteristics are

- 1. First Normal Form (1NF):** A Relation is said to be in first normal form if it is already in un normalized form and it has no repeating group.
- 2. Second Normal Form (2NF):** A Relation is said to be in second normal form if it is already in first normal form and it has no partial dependency.
- 3. Third Normal Form (3NF):** A Relation is said to be in third Normal form if it is already in second normal form and it has no transitive dependency.
- 4. Boyce code Normal Form(BCNF):** A Relation is said to be in Boyce code Normal form if it is already in third normal form and every determinant is a candidate key.
- 5. Fourth Normal Form (4NF):** A Relation is said to be in fourth normal form if it is already in Boyce code normal form and it has no multi valued dependency.
- 6. Fifth Normal Form(5NF):** A Relation is said to be fifth normal form if it is already in fourth normal form and it has no loss less decompose.

Eg: Normalization of construction company Report

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS	Total_Charge
15	Evergreen	103	June E .Arbough	Elect.Engineer	84.50	23.8	2011.1
		101	Swetha	Database designer	105.00	19.4	2037
		105	Lakshmi	Database designer	105.00	35.7	3748.5
		106	Durga	Programmer	35.75	12.6	450.45
		102	Ram	Systems Analyst	96.75	20.8	2012.4
SubTotal							10259.45
18	Amber wave	114	Harika	Applications designer	48.10	25.6	1231.36
		118	Ganesh	General support	18.36	45.3	831.708
		104	Sri	Systems analyst	96.75	32.4	3134.7
		112	Hari	DSS Analyst	45.95	44.0	2021.8
Sub Total							7219.568
22	Rolling Tide	105	Sruthi	Database designer	105.00	64.7	6793.5
		104	Raju	Systems analyst	26.75	48.4	1294.7
		113	Ravi	Application designer	48.10	23.6	1135.16
		111	Ramesh	Clerical support	26.87	22.0	591.14
		106	Rao	Programmer	35.75	12.8	457.6
Sub Total							10272.1
25	Starflight	107	Rekha	Programmer	35.75	24.6	879.45
		115	Rani	System analyst	96.75	45.8	4431.15

		101	John	Database designer	105.00	56.3	5911.5	
		114	Manikanta	Applications designer	48.10	33.1	1592.11	
		108	Nalini	System analyst	96.75	23.6	2283.3	
		118	James	General secretary	18.36	30.5	559.98	
		112	P J	DSS Analyst	45.95	41.4	1902.33	
							Sub Total	17559.82
							Total Amount	45310.94

The construction company report is represented in the form of relation. The relation named as CONSTRUCTION_COMPANY this is in un normalized form as shown below

CONSTRUCTION_COMPANY(Proj_No,Proj_Name,(Emp_No,Emp_Name, Job_Classification, Charge_Per_Hour, Hours_Billed)) -----> (1)

The field Total charge, SUB TOTAL,GRAND TOTAL are not included in the relation because they are derived Attribute.

First Normal Form:

In Relation (1), the fields in the inner most set of parenthesis put together is known as repetating group. This will result in redundancy of data for the first two relations remove the repetating group. Hence the relation 1 is subdivided into two relations to remove repeating group

PROJECT(proj_No,proj_Name) -----> (2)

PROJECT_EMP(Proj_No, Emp_No,Emp_Name,Job_Class,Charge_Per_Hour,Hours_Billed) ----- →(3)

Now above relation (2) & (3) are in 1NF. In relation (3) Proj_No , Emp_No jointly serve as key field.

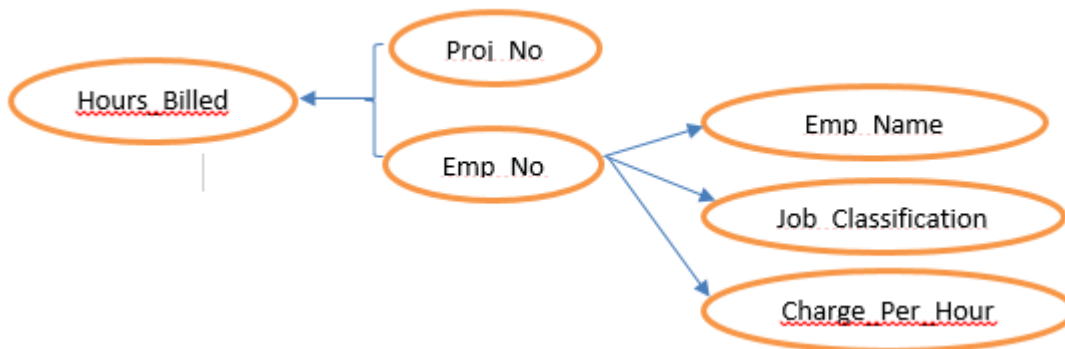
Second Normal Form:

Definition of Partial Dependency:

Non key attribute are depending on the part of the composite primary key then it is said to be Partial Dependency.

In Relation 2 the number key fields is only one and hence there is no scope for partial dependency the absence of partial dependency in relation 2 takes it 2NF without any modification.

The dependency diagram of relation 3 is shown below



In the above diagram **Hours_Builled** depends on Project_No and Emp_No but the remaining non key fields (Emp_Name,Job_Class,Charge_per_Hour)depends on Emp_No this situation is an example of 2nd normal form .Hence the relation 3 is divided into 2 relations.

Assignment(Proj_No, Emp_No, Hours_Billed)----->(4)

Emp_Job(Emp_No,Emp_Name,Job_Class,Charge-Per_Hour)----->(5)

The Relations (4) & (5) are in 2NF

Third Normal Form:

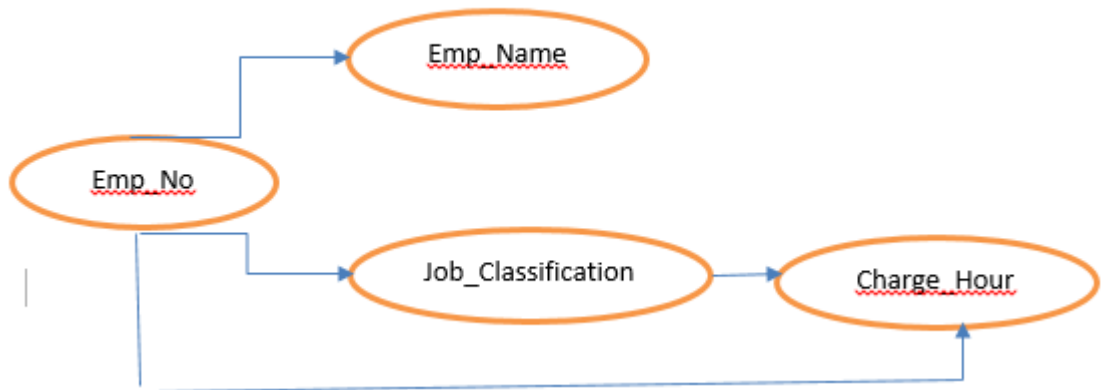
Transitive Dependency:

If one non prime attribute is determines the other non prime attribute then it is called as transitive dependency..

In Relation (2) there is only one non key field. This means that it has no transitive dependency. Hence Relation (2) can be treated as 3NF without any modification similarly in relation (4) there is only one non key field. This means that it has no transitive dependency. Hence relation (4) can be treated as 3Nf without any modification.

In Relation(5) Charge_Per_Hour depends on Job_Classification this means that relation (5) has transitive dependency. The dependency diagram for the relation (5) is shown below.

Diagram



Hence relation (5) is sub divided into two relations. Relation (6) and relation (7) as shown below.

Job(Job_Class, Charge_Per_Hour)-----→(6)

Emp(Emp_No,Emp_Name,Job_Class)-----→(7)

For a practical application it is sufficient to normalized up to either 2NF or 3NF

Hence, the process of normalization is stopped and the final 3NF relations of construction company as shown below.

Project(Proj_No,Proj_Name)-----→(1)

Assignment(Proj_No, Emp_No, Hours_Billed)-----→(2)

Emp(Emp_No,Emp_Name,Job_Classification)-----→(3)

Job(Job_Classification, Charge_Per_Hour)-----→(4)

Improving Design:

How to improve the design of the database?

1. Evaluate primary key Assignment:

Each time a new employee is entered into the employee table, A job class value must be entered. Unfortunately the data entry in the job class contains an error, that lead to referential integrity violation. For example entering "DB Designer" instead of Database Designer for the Job_class attribute in the Employee table will trigger such a violation . Therefore it is better to add job code attribute in job relation and employee relation.

Emp(Emp_No,Emp_Name,Job_Code)

Job(Job_Code,Job_Classification,Charge_Per_Hour)

2. Evaluate Naming conversions:

In the job relation job classification will be changed to Job_Description and Charge_Per_Hour will be changed to Job_Chrg_Hour. In the assignment relation Hours_Billed will be changed to Assign_Hours_Billed.

Job(Job_Code,Job_Description,Job_Chg_Hour)
Assignment(Proj_No,Emp_No,Assign_Hours_Billed)

3. Refine Attribute Atomicity:

An Automatic Attribute is one that cannot be further sub divided such an attribute is said to be automaticity. In employee table the attribute Emp_Name is not a automaticity because it is further sub divided as Emp_LName,Emp_FName,Emp_Init. These attributes are added to the employee table.

Emp(Emp_No,Emp_LName,Emp_FName,Emp_Init,Job_Code)

4. Identify New Attribute:

If the employee table used in real world environment several other attributes will be added. For example Social_Security_Number, Date_of_Joining, Date_of_Birth,Hire_Date,Gross_Salary,Net_salary etc will be added to improve relation.

Emp(Emp_No,Emp_LName,Emp_FName,Emp_Init,Hire_Data,Gross_Salary,Job_Code)

5. Identify New Relationship:

When we create a new relationship between the table it will not produce unnecessary duplication. Then we create a new relationship.

6. Refine primary Keys:

The combination of Emp_No and Proj_No is a primary key in the Assignment table. For example if we add a assigned hours more than one time for a particular project then it violates the primary key constraints.

To avoid the violation to add additional attribute Assign_Date to the assignment table. If we want to add assigned hours for a particular project

more than one time in the same day then it will violates the primary key constraints. The same data entry gives no problem when Assign_No is used as a primary key in the Assignment relation.

Assignment(Assign_No,Assign_Date,Proj_No,Emp_No,Assig_Hour_Billed)

7. To Maintain Historical Accuracy:

It is assumed that the Job_Chg_Hour will change over time. The changes to each project were billed by multiplying the hours worked on the project in the assignment table by the Job_Chg_Hour in the job table. Those changes would always show the current **change_per_hour** stored in the job table rather than the job charge hour that was in effect at the time of assignment. Because of that we are adding an attribute **Assign_Chg_Hour** to the Assignment table.

Assignment(Assign_No,Assign_Date,Proj_No,Emp_No,Assig_Hour_Billed,Assign_Chg_Hour)

8. Evaluate using Derived Attribute:

The derived attribute Assign_Charge is added to the Assignment relation and Assign_charge is updated by multiplying with Assign_Chg_Hour with the Assign_Hours_Billed. However the derived attribute Assign_Charge in the Assignment table makes easy to write reports or invoices.

Boyce code Normal Form(BCNF):

The BCNF can be violated only when the table contains more than one candidate key

Candidate key:

A key is said to be candidate key if the superkey that does not contain a subset of attributes i.e the key itself a superkey.



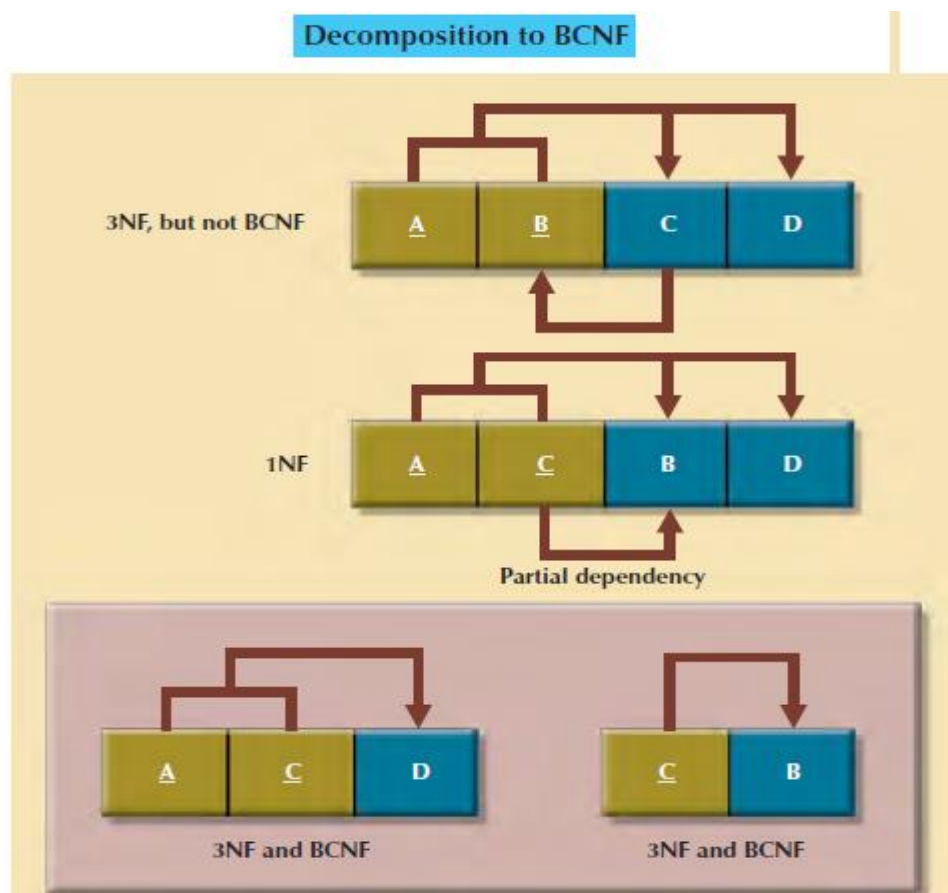
These functional dependencies are shown below.

$$\underline{A+B} \text{ -----} \rightarrow C,D$$

$$\underline{C} \text{ -----} \rightarrow B$$

The table structure has no partial dependency and there is no transitive dependency. But the condition $C \text{ ---} \rightarrow B$ indicates that a nonkey attribute determines the part of key the primary key, causes the table to fail to meet the BCNF requirements.

To convert the above table structure from 3NF to BCNF, first change the primary key to A+C. The dependency $C \text{ ---} \rightarrow B$ means that C is in effect a superset of B. The Decomposition procedures to produce the results shown below.



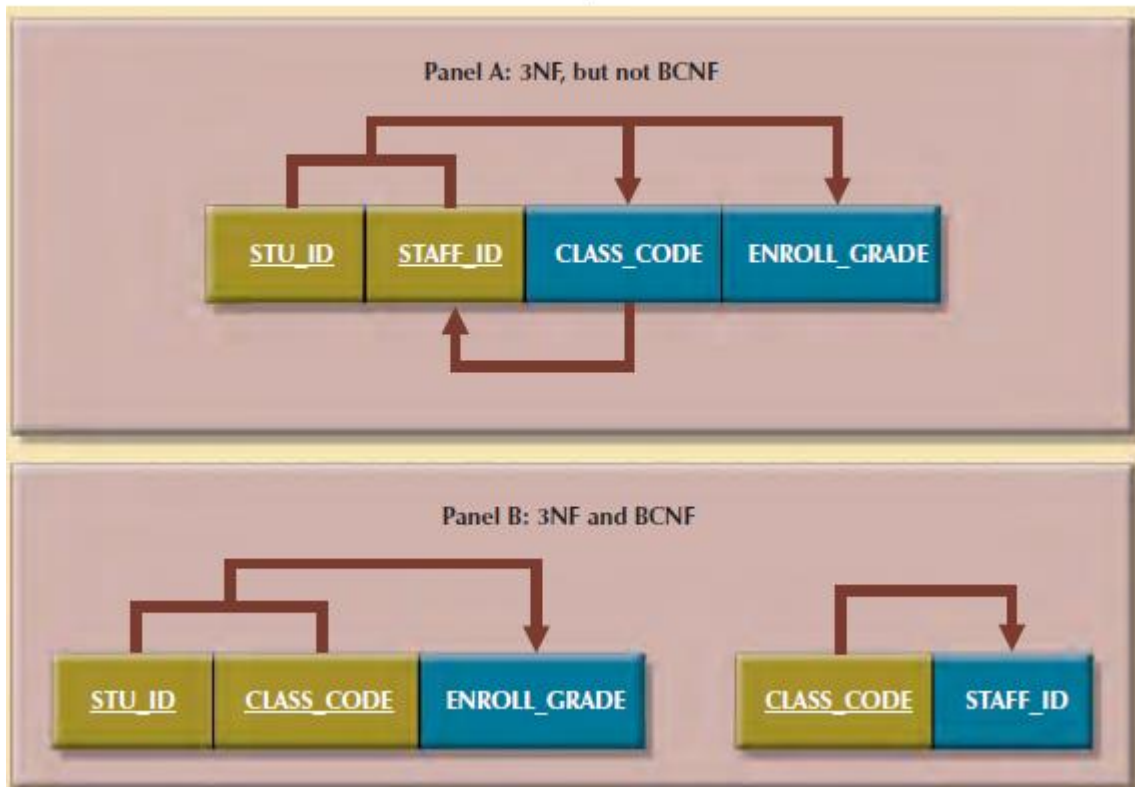
Example for BCNF:

Each Class_Code identifies a class iniquely. A student contains many classes and earning the grades respectively. A staff member can teach many classes. But each class is taught by only one staff member.

Sample Data for a BCNF Conversion

STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32456	B

BCNF Decomposition



Stu_Id+Staff_Id -----→Class_Code, Enroll_grade

Class_Code -----→Staff_Id

The above table contains two candidate keys to violates the BCNF. Now we can eliminate the one candidate key from the above table.

Fourth Normal Form (4NF):

Consider an employee can have multiple assignment i.e, that employee works as an volunteer in service organization and worked in different projects which is shown below

Tables with multivalued dependencies

Database name: Ch06_Service

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	1
10123	UW	3
10123		4

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	
10123	UW	
10123		1
10123		3
10123		4

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	1
10123	RC	3
10123	UW	4

The above contains two sets of independent multi valued dependencies (i.e., Org_Code, Proj_Code). If volunteer-1 and Volunteer_2 two tables are implemented. The two tables contains null values.

In volunteer-3 table has a primary key but it is composed of all attributes of the table. When you consider like this it produces many redundancies.

To eliminate multi valued dependency by creating the assignment and service tables as shown below.

PROJ_CODE	PROJ_NAME	PROJ_BUDGET
1	BeThere	1023245.00
2	BlueMoon	20198608.00
3	GreenThumb	3234458.00
4	GoFast	5674000.00
5	GoSlow	1002500.00

ASSIGN_NUM	EMP_NUM	PROJ_CODE
1	10123	1
2	10121	2
3	10123	3
4	10123	4
5	10121	1
6	10124	2
7	10124	3
8	10124	5

EMP_NUM	EMP_LNAME
10121	Rogers
10122	O'Leary
10123	Panera
10124	Johnson

ORG_CODE	ORG_NAME
RC	Red Cross
UW	United Way
WF	Wildlife Fund

EMP_NUM	ORG_CODE
10123	RC
10123	UW
10123	WF

In the Assignment table and service table does not contain multi valued dependency

De Normalization: (Under Construction..Ravindra)

Normalization is a very important in database design. Generally the higher normal forms the more the relational join operations required to produce a specific output. A successful design must also consider end user requirement for fast

performance. Therefore occasionally we expected to de normalize some portions of the database design in order to meet performance require.

De Normalization produces lower normal forms 3Nf will be converted to 2NF or 2Nf will be converted to 1NF.

Eg: The need for de normalization due to generate evaluation of faculty report in which each row list the scores of obtaining during the last 4 semester taught.

Faculty Evaluation Report:

instruct or	Dep t.	Se m-1	Mea n	Se m-2	Mea n	Se m-3	Mea n	Se m-4	Mea n	Last_se m avg

We can generate easy above the report but the problem arises. The data are stored in a normalized table. In which each row represented a different score for a given faculty in a given semester.

EVALDATA:

ID	Instructor	DEPT.	Mean	Semistor

It is some difficulty to generate faculty evaluation report the normalized table

The other table FACHLST faculty history table contains the last four semester mean for each faculty .The faculty history table is a temporary table created from the evaldata as shown below.

Instruct or	Dep t.	Se m-1	Mea n	Se m-2	Mea n	Se m-3	Mea n	Se m-4	Mea n	Last_se m avg

The FACHIST is a un normalized from table using the table we can generate .The faculty evaluation report very firstly. After generating the report, the temporary table, FACHIST will be deleted. We are doing like this, we can increase the performance of the database

Chapter – 6

Advanced Data Modeling

Advanced Data Model:

The Extended Entity Relationship Model (EERM) sometimes referred to as the enhanced entity relationship model because adding more semantic constructs to the original entity relationship model. The ER Model constructs Entity super types, entity sub types and entity clustering.

Entity Super types, Entity Sub types:

In an Organization contains different types of employee and all the employees are not having the same attributes. If you create a one table for all employees to store their information many columns have null values.

For Example the pilot shares certain characteristics with other employee such as ***employee_no, emp_name, emp_address, employee_hire_date*** on the other employees. But pilot characteristics are not shared by other employee. The pilot characteristics are ***employee_license and employee_rating*** will generates nulls for employees who are not pilot.

The pilot entity store only the attributes that are unique to pilot, and the employee entity store attribute that are common to all employees.

We can conclude that pilot is a *sub type* of employee and employee is a *super type* of pilot. An entity super type is a generic type i.e, related to one or more entity subtypes, where the entity super type contains common characteristics entity subtype contains unique characteristics.

Specialization Hierarchy:

Entity super types and sub types are organized in a hierarchy which describes the higher level entity super types (parent entity) and lower level entity (child entity) sub types.

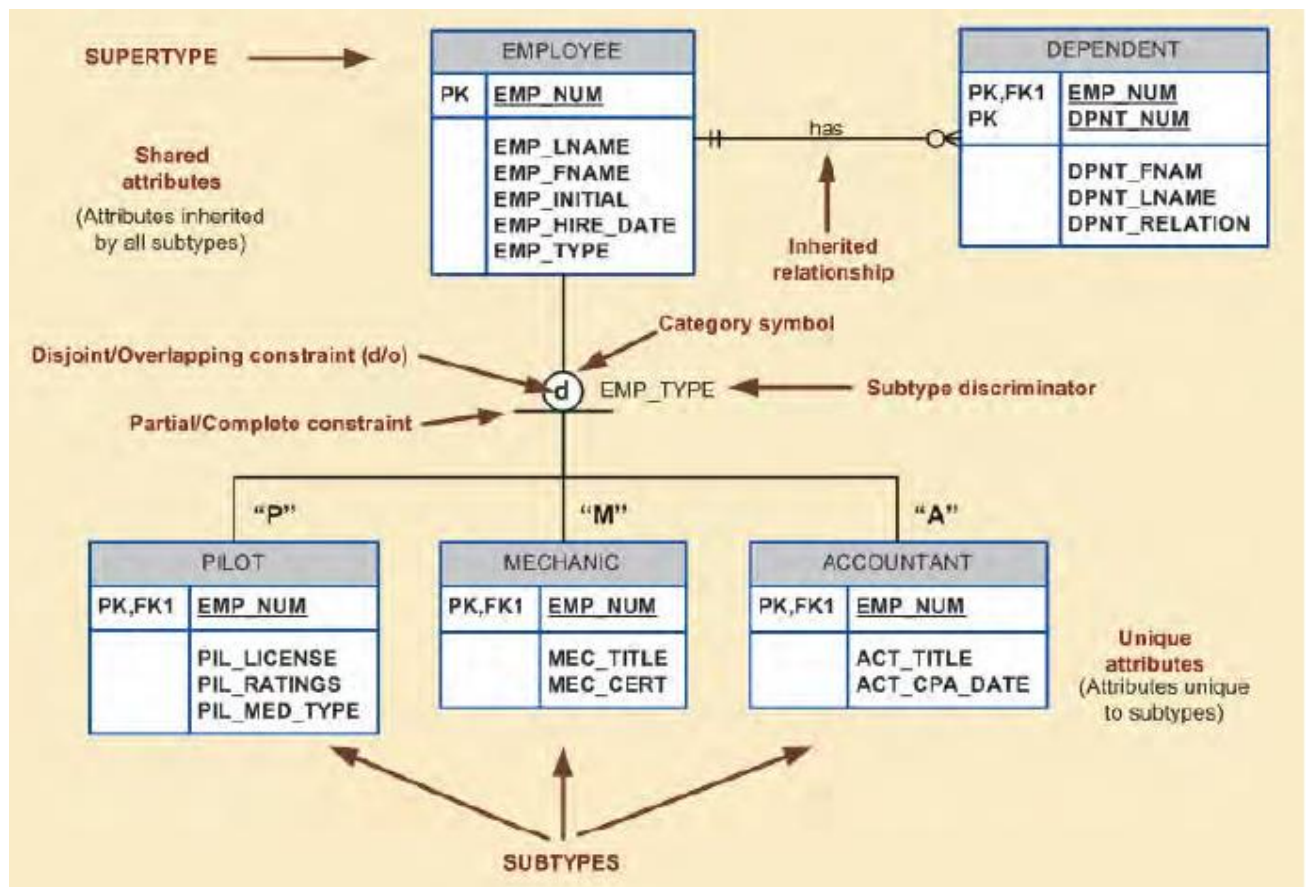
The following diagram shows specialization hierarchy by an employee super type and three entity sub types.

1. Pilot
2. Mechanic
3. Accountant.

The specialization hierarchy reflects one-to-one relationship between super entity type and sub entity type.

Eg:

1. A pilot sub type is related to one instance of employee super type.
2. A mechanic sub type is related to one instance of employee super type.
3. An Account sub type is related to one instance of employee super type.



Inheritance:

The property of inheritance enables an entity subtype, to inherit the attributes and relationships of the super type.

In the above example the **employee entity** super type participating in a one-to-many relationship with a **dependent entity** through inheritance all subtypes also participated in that relationship.

Subtype Discriminator:

A sub type discriminator is the attribute in the super type entity that determines to which sub type is related.

In the above example if the sub type discriminator is emp_type.

If the emp_type has a value of p the super type is related to pilot sub type.

If the emp_type has a value of A the super type is related to Account subtype.

If the emp_type has a value of M the super type is related to mechanic subtype.

Disjoint & Overlapping Constraints:

An entity super type can have disjoint or overlapping entity supertypes.

Disjoint subtype are sub types that contains a unique subset of the super type entity set.

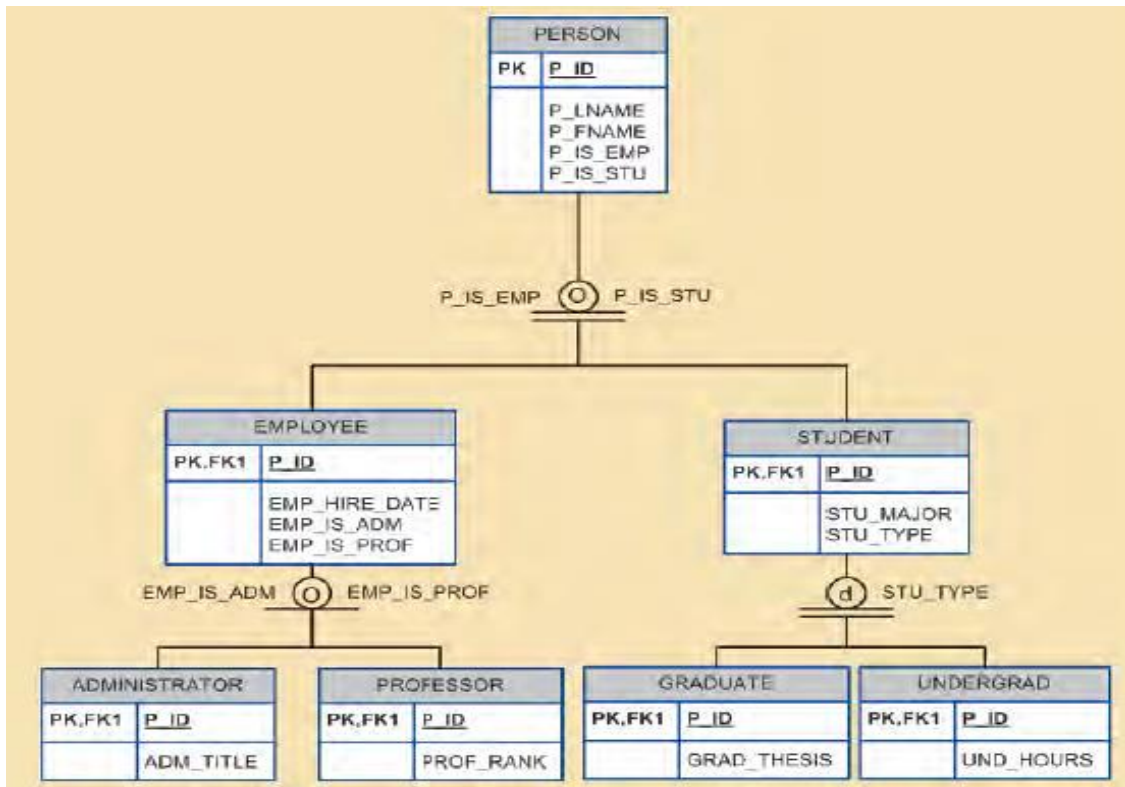
The Disjoint subtypes are indicated by the letter 'd' inside the category shape. In disjoint the super type entity is related to only one sub entity types.

Eg: An employee who is a pilot can appear only in the pilot sub type, not in any of other sub types.

Overlapping subtypes are subtypes that contains non unique subsets of the super type entity set.

The Overlapping subtypes are indicated by the letter 'o' inside the category shape. In the overlapping the super type entity is not relate to only one sub entity types.

Eg: An employee may be a professor as well as administrator.





The Administrator and professor overlap the super type entity employee.

Completeness constraint:

The completeness constraint can be partial or total.

Partial completeness means that **not every super type entity** is a member of sub type entity. A single horizontal line under the circle represents a partial constraint O.

Total completeness means that every super type is must be a member of **at latest one sub type**. A double horizontal line under the circle represents the total completeness constraint.

TYPE	DISJOINT CONSTRAINT	OVERLAPPING CONSTRAINT
Partial 	Supertype has optional subtypes. Subtype discriminator can be null. Subtype sets are unique.	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique.
Total 	Every supertype occurrence is a member of a (at least one) subtype. Subtype discriminator cannot be null. Subtype sets are unique.	Every supertype occurrence is a member of a (at least one) subtype. Subtype discriminators cannot be null. Subtype sets are not unique.

Specialization and Generalization:

We can use various approaches to develop entity super types and sub types.

Specialization is the top down process of identifying lower level entity sub type from a higher level entity super type.

Eg: The specialization is used to identify multiple entity supply (Pilot, Mechanic, Accountant) from the super entity employee.

Generalization is the bottom-up process of identifying higher level entity super types from a lower level entity sub types.

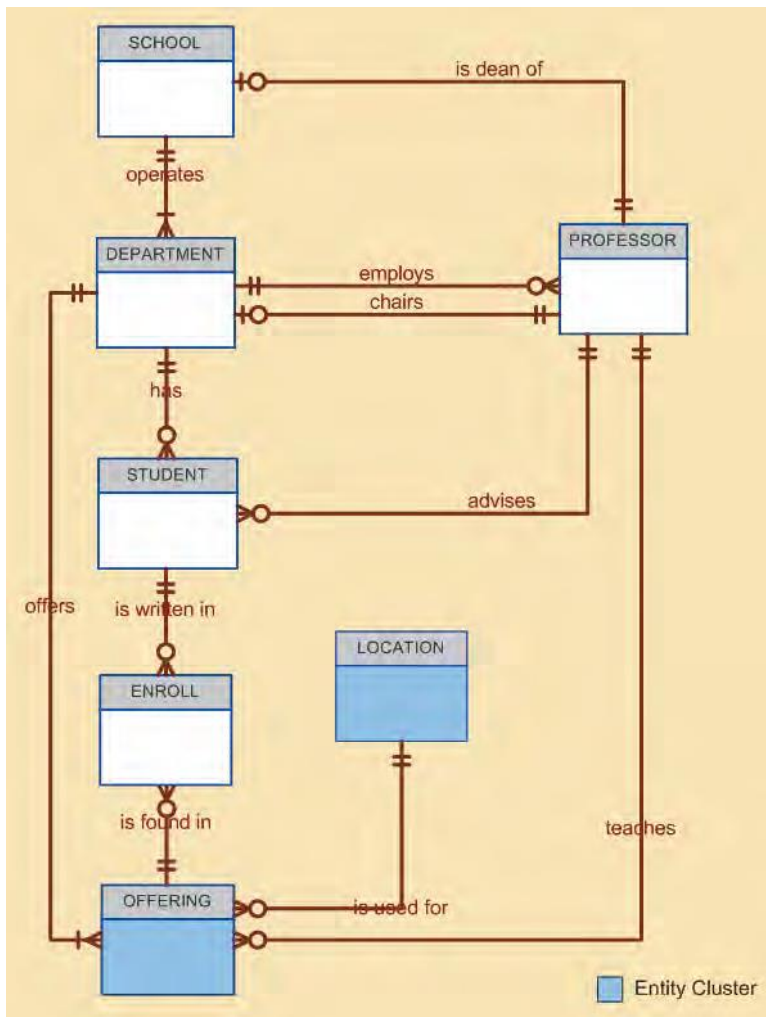
Eg: The Generalization is used to identify entity super type from the sub type (Pilot, mechanic, Accountant).

Entity Clustering:

Generally the data model will develop an initial ERD containing a few entities. As the designed approach completion the ERD will contain hundreds of entities and relationships. In those cases, we can use entity cluster to minimize the number of entities in the ERD.

An entity cluster is a virtual entity type used to represent multiple entities and the relationship in the ERD.

An entity cluster is formed by combining multiple inter related entities into a single entity object. An entity cluster is considered virtually in the sense that it is not actually an entity in the final ERD.



Entity Integrity:

The most important characteristics of an entity is its primary key, which uniquely identifies each entity instance. The primary key and foreign key works together implement relationship between the tables in the relational data model.

Natural keys & Primary keys:

The unique identifies is commonly uncouted in the real world. For example **class_no** to register for classes, **invoice_no** to identify a particular invoice, **account_no** to identify credit cards and soon. These Examples contains natural keys.

The natural keys class_no, invoice_no, account_no is used to uniquely identify the real world objects.

If an entity has a natural identifier, a data modeler uses that natural key as the primary key of the entity.

Primary key Guidelines:

The primary key main function is to uniquely identify an entity for a given primary key value the relational model can determine values of all dependent attributes.

The second function is primary key and foreign key implement relationship between tables or entities.

Characteristics of Primary Key:

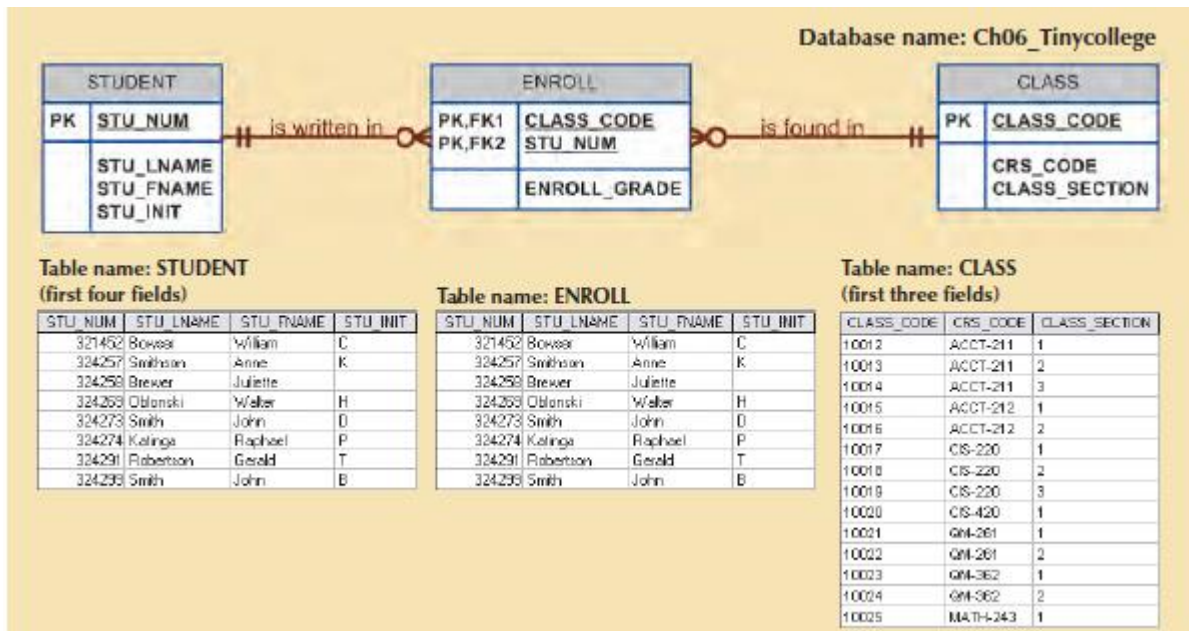
- A primary key contains unique values and not accept the null values.
- The primary key should be permanent and unchangeable
- A primary key should have the minimum number of attributes.
- Unique values can be managed when primary keys are numeric

When to use composite primary keys:

The composite primary keys are particularly useful in two cases.

1. As Identifier of composite entities.
2. As Identifier of weak entities.

In the first case assume that we have a student entity and class entity and the relationship between these two entities is many to many via enroll entity. The enroll entity contains key fields of student entity and class entity which is used to identify entity instance in the enroll entity.



In the 2nd case a weak entity in a strong relationship with a parent entity is normally used.

Eg: The key field of employee entity is used one of the key filed of dependent entity.

Emp(Emp_no, Emp_Fname,Emp_Lname,email)

Dependent(Emp_no, Depn_ Depn_Fname, Depn_Lname, Depn_Addr)

When to use surrogate primary keys?

These are some instances when a primary key does not exist in the real world object.

(or)

When the existing natural key is not suitable as primary keys.

For Example: Consider the facility that rent for rooms for small parties. The manager of the facility keep the all the events in the following table formats.

Date	Time_start	Time_End	Room	Event_name	Party-of

The above table can be represented as Event entity

EVENT(Date, Time_start, Time_End, Room, Event_name, Party-of)

In the above entity there is no simple natural keys i.e, used as a primary key in the model.

Based on the concept of primary key we suggest one of these options (Date, Time_start, Room) or (Date, Time_End, Room).

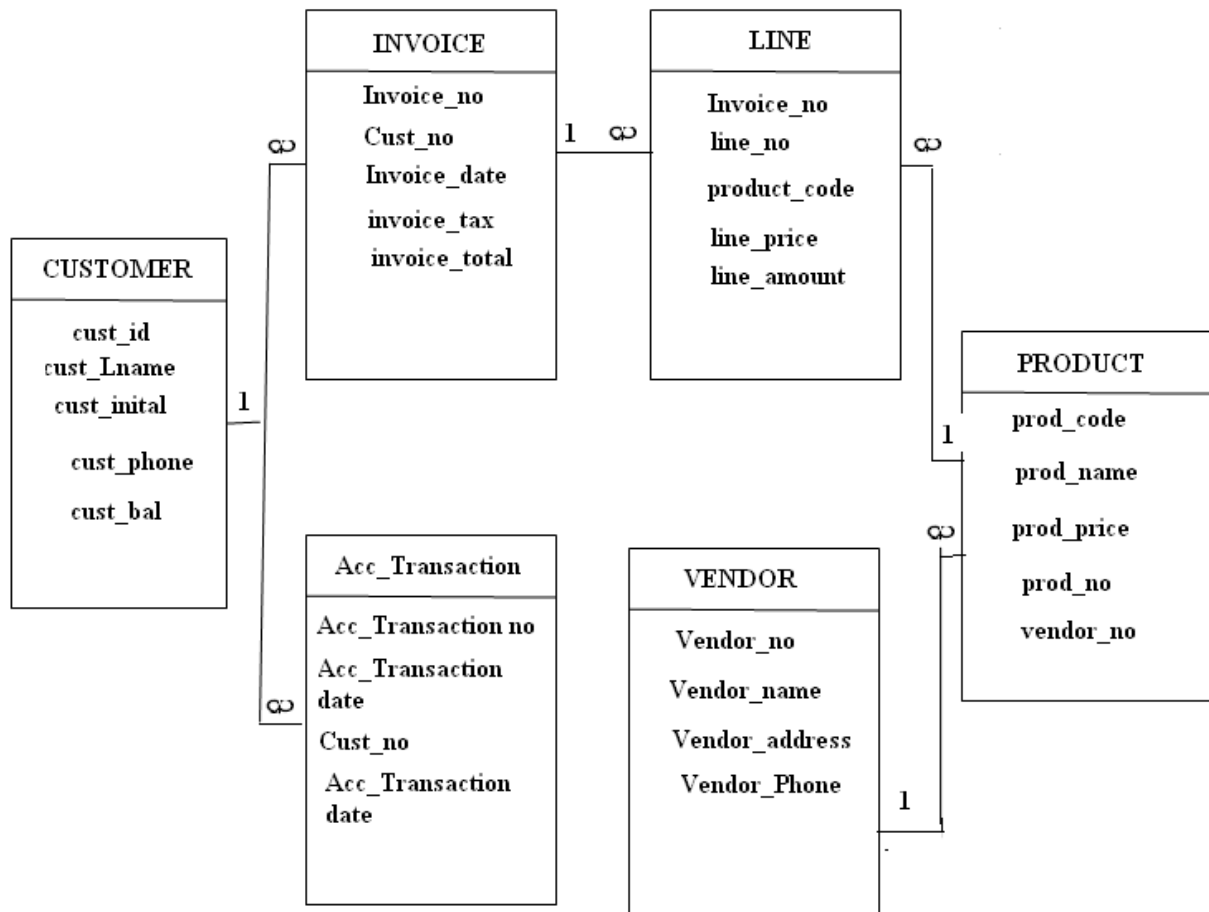
1. When Implementation of data model, the composite primary key in the event entity makes complexity and also coding.
2. The solution to the problem is to use a numeric single attribute as surrogate primary key.

EVENT(Date, Time_start, Time_End, Room, Event_name, Party-of)

Transaction Control and Concurrency Control

Transaction:

A Transaction is a series of actions to be performed on the database such that either all actions are performed or none.



Suppose that we sell a product to a customer, our sales transaction consists of at least the following parts.

1. We must write a new customer invoice.
2. We must reduce the quantity on hand in the product entities.
3. We must update the Acc_Transaction.
4. We must update the customer_bal.

In database terms a transaction is any action that read from and/or writes to a database. A transaction may consists of a single SQL statement or a series of related update statements or insert statements or combination of select, update and insert statements.

1. A Transaction is logical units of work that must entirely complete or entirely aborted no intermediate states are accepted. All of the SQL statements in a transaction must be completed successfully. If any of the SQL statements fail the entire transaction is roll back.
2. A successful transaction changes the database one consistent state to another. A consistent database state is one in which all data integrity constraints are satisfied.
3. Each database request generates several I/O operations that reads from or writes to physical storage medium.

Transaction Properties

The Transaction has the following properties.

- Atomicity.
- Consistency.
- Isolation.
- Durability.

Atomicity: All Operations of the transaction must be completed. If not the transaction is aborted.

Example: If a transaction T_1 has four SQL requests, all four requests must be successfully completed otherwise the entire transaction is aborted.

Consistency: When a transaction is completed, the database must be consistent state. That means it must satisfies all integrity constraints otherwise the entire transaction is aborted.

Isolation: The data used during the execution of the transaction cannot be used by a second transaction until the first transaction is completed.

Durability: Once the transaction changes are done they cannot be undone even in the system failure.

Transaction Management in SQL

When a transaction sequence is initiated by a user the sequence must continue through SQL statements until one of the following four events occurs.

1. A COMMIT statement is reached, in which case all changes are permanently recorded into the database.
2. A ROLLBACK statement is reached in which case all changes are aborted and the database is ROLLBACK to previous state.

3. The END of the transaction is successfully reached. In which case all changes are permanently recorded within the database. This action is equivalent to COMMIT.
4. The transaction is abnormally terminated. In which case the changes made in the database are aborted and the database is ROLLBACK to previous state. This action equivalent to ROLLBACK.

The Transaction Log

1. The DBMS uses a transaction log file to keep track of all transaction that update the database.
2. The Information stored in the log file is used by the DBMS for ROLLBACK, abnormal termination or system failure.
3. While DBMS executes transactions that modify the database, it also automatically updates the transaction log.
4. The transaction log stores
 - a. A record for the beginning of transaction.
 - b. For each transaction components
 - i. The type of operation is being to perform.
 - ii. The name of the object effected by the transaction (Name of the table).
 - iii. The "Before" and "After" values for the fields being updated.
 - iv. Pointers to the previous and next transaction log entries for the same transaction.
 - c. The end of the transaction log.

Eg: update product set prod_qua=prod_qua-2

Where prod code='p1001';

Update Customer set cus_bal=Bal+8000

Where cust_no='c1234';

The transaction log for the above transaction is,

TRL_id	TRX	PREV -PTR	NEXT -PTR	OPERATION	TABLE	ROW-ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE

	NUM								
341	101	Null	352	START	Start operation				
352	101	341	363	update	Product	1011	Product_poh	25	23
363	101	352	365	Update	Custmer	363	Cust_Balance	525.15	615.75
365	101	363	null	commit	End of operation				

If the system failures occur the DBMS will examine the transaction log for incomplete transaction and ROLLBACK the database to its previous state.

Q. Explain Concurrency control in transaction management.

Concurrency control is important because the simultaneously execution of transaction over a shared database can create several data integrity and consistency problems.

1. Lost updates.
2. Uncommitted data
3. Data inconsistency.

Lost Updates:

The Lost update problem occurs when two concurrent transitions T_1 and T_2 are updating the same data element and one of the update is lost.

Eg: The two concurrent transactions T_1 and T_2 update the **prod_qua** value for same item in the prod table. Assume that current **prod_qua** value is 35.

Transaction	Computation
T_1 : Purchase 100 units	$prod_qua = prod_qua + 100$
T_2 : Sale 30 units	$prod_qua = prod_qua - 30$

The following table shows the serial execution of those transaction under the normal circumstance gives the answer.

Time	Transaction	Step	Stored Value
1	T1	READ PROD_QOH	35
2	T1	PROD_QOH=35+100	
3	T1	WRITE PROD_QOH	135
4	T2	READ PROD_QOH	135
5	T2	PROD_QOH=135-30	
6	T2	WRITE PROD_QOH	105

Suppose that transaction is read a prod_qua value from a table before a previous transaction has been committed. This sequence shows the following table, how the lost update problem can occur.

Time	Transaction	Step	Stored Value
1	T1	READ PROD_QOH	35
2	T2	READ PROD_QOH	35
3	T1	PROD_QOH=35+100	
4	T2	PROD_QOH=35-30	
5	T1	WRITE PROD_QOH	135
6	T2	WRITE PROD_QOH	5

Uncommitted Data

The uncommitted data problem occur when two transactions T_1 and T_2 are executed concurrently and the first transaction T_1 is Rolled back after the second transaction T_2 has already accessed the uncommitted data.

Eg: The two concurrent transactions T_1 & T_2 update the prod_qua value for same item in the prod table assumes that the current prod_qua value is 35.

Transaction	Computation
T_1 : Purchase 100 units	$\text{prod_qua} = \text{prod_qua} + 100$
T_2 : Sale 30 units	$\text{prod_qua} = \text{prod_qua} - 30$

The following table shows under normal circumstance, the serial execution of this transaction use that transaction.

Time	Transaction	Step	Stored Value
1	T1	READ PROD_QOH	35
2	T1	PROD_QOH=35+100	
3	T1	WRITE PROD_QOH	135
4	T1	ROLLBACK	35
5	T2	READ PROD_QOH	35
6	T2	PROD_QOH=135-30	
7	T2	WRITE PROD_QOH	5

The following table shows, how the uncommitted data problem can arise when the Roll back is completed after T₂ has begin its execution.

Time	Transaction	Step	Stored Value
1	T1	READ PROD_QOH	35
2	T1	PROD_QOH=35+100	
3	T1	WRITE PROD_QOH	135
4	T2	READ PROD_QOH (Read uncommitted data)	135
5	T2	PROD_QOH=135-30	
6	T1	ROLLBACK	35
7	T2	WRITE PROD_QOH	105

Inconsistency Retrievals

Inconsistency retrieval occur when a transaction access data before and after another transaction finish working with same data.

For example the transaction T_1 calculates the total `prod_qua` of the products stored in the product table. At the same time T_2 updates `prod_qua` for two products in the product table.

TRANSACTION 1	TRANSACTION 2
Select sum(Prod_QOH)from Product	Update Product set PROD_QOH+10 where Prod_code=1003; Update Product set PROD_QOH- 10 where Prod_code=1004; Commit;

The following table show the results of above two transactions.

Prod_code	BEFORE	AFTER
	PROD_QOH	PROD_QOH
1001	8	8
1002	32	32
1003	15	15+10=25
1004	23	23-10=13
1005	8	8
1006	6	6
	Total=92	Total=92

The final result shows in the above table are correct.

The following demonstrates that inconsistency retrievals are possible. During the transaction T₁ executes in the absence of concurrency control.

While summing the prod_qua of the transaction **T₁ reads the afterprod_qua 25** for prod_code=1003 and reads the **before prod_qua=23** for prod_code=1004. Gives the result 102.

The following table shows the inconsistent retrievals'.

TIME	TRANSACTION	STEP	VALUE	TOTAL
1	T1	READ PROD_QOH for PROD_CODE=1001	8	8
2	T1	READ PROD_QOH for PROD_CODE=1002	32	40
3	T2	READ PROD_QOH for PROD_CODE=1003	15	
4	T2	READ PROD_QOH =15+10		
5	T2	WRITE PROD_QOH for PROD_CODE=1003	25	
6	T1	READ PROD_QOH for PROD_CODE=1003	25	65(AFTER)
7	T1	READ PROD_QOH for PROD_CODE=1004	23	88(Before)
8	T2	READ PROD_QOH for PROD_CODE=1003	23	
9	T2	READ PROD_QOH =23-10		
10	T2	WRITE PROD_QOH for PROD_CODE=1004	13	
11	T2	COMMIT		
12	T1	READ PROD_QOH for PROD_CODE=1005	8	96
13	T1	READ PROD_QOH for PROD_CODE=1006	6	102

The Shedular

The shedular is a special DBMS process. The shedular uses the concurrency control algorithms such as locking or time stamp methods to control the concurrent executions of transactions on the same database.

Concurrency Control with Locking methods

A transaction acquires a lock before to data access the lock is released when the transaction is completed. So that another transaction can lock the data item for its exclusive use all lock information is managed by a lock manager.

Lock Granularity

Lock granularity indicates the level of lock use. Locking can takes place at the following levels:

1. Database level lock
2. Table level lock
3. Page level lock
4. Row level lock
5. Field level lock.

Database level:

In Database level lock , the entire database is locked, preventing the use of any tables in the database by transaction T_2 while transacting T_1 is being executed.

1. This type of locking is unsuitable for multi user DBMS because thousands of transactions waiting for the previous transactions to be completed.
2. In database level lock transactions cannot access the same database concurrently even when they use different tables.

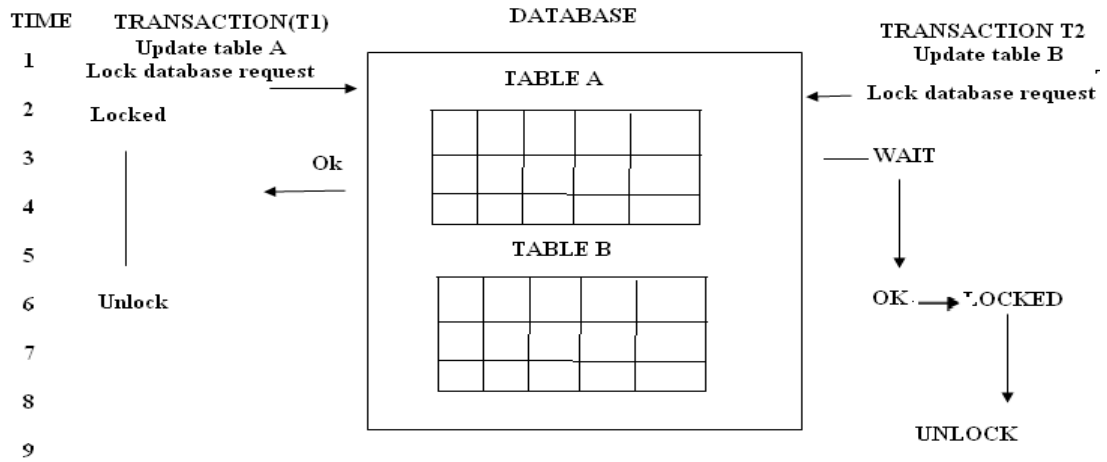
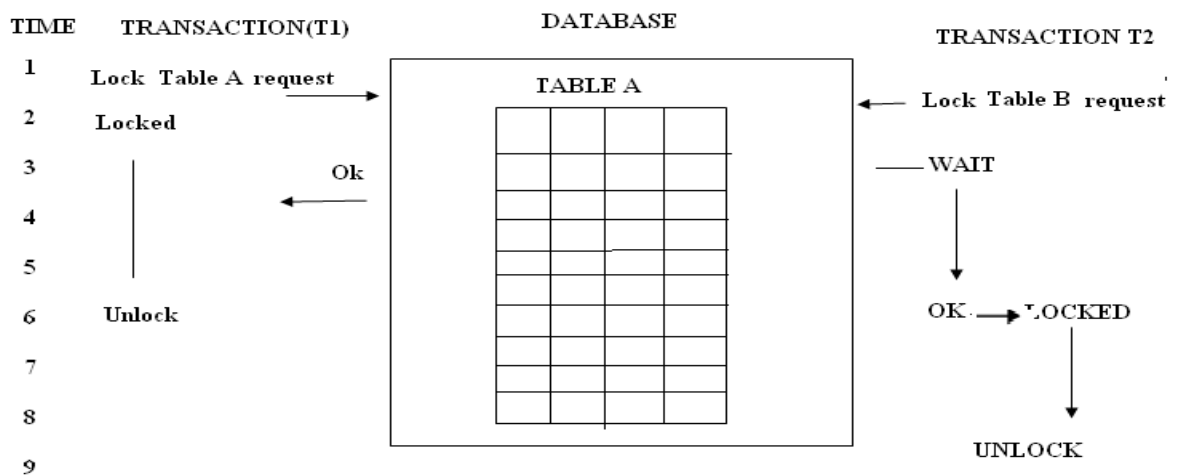


Table Level Lock

1. In table level lock, the entire table is locked preventing access to any row by transaction T₂ while transaction T₁ is using a table.
2. If a transaction requires several tables, each table may be locked. Two transactions can access the same database as long as they access different tables.
3. Table level locks also cause traffic jam when many transactions are waiting to access the same table.

The following shows transactions T₁ and T₂ cannot access the same table even when they trying to use different rows, T₂ must wait until T₁ unlocks the table.

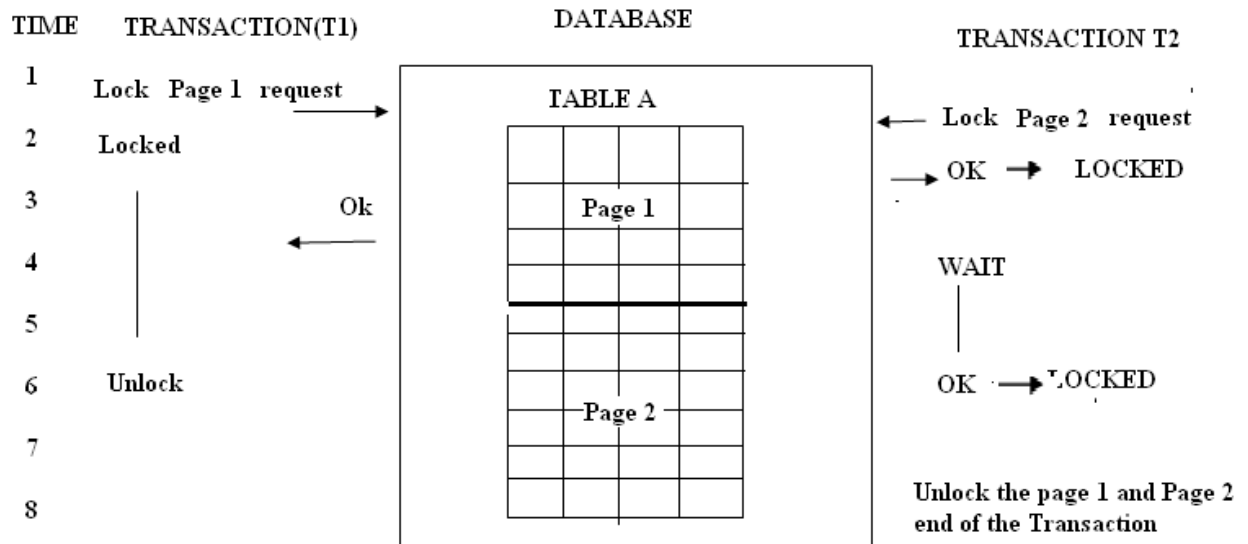


Page Level

A page has a fixed size such as 4KB, 8KB or 12KB.

1. A table can span several pages and a page can contains several rows. A page level locks are most frequently used multi user DBMS locking method.

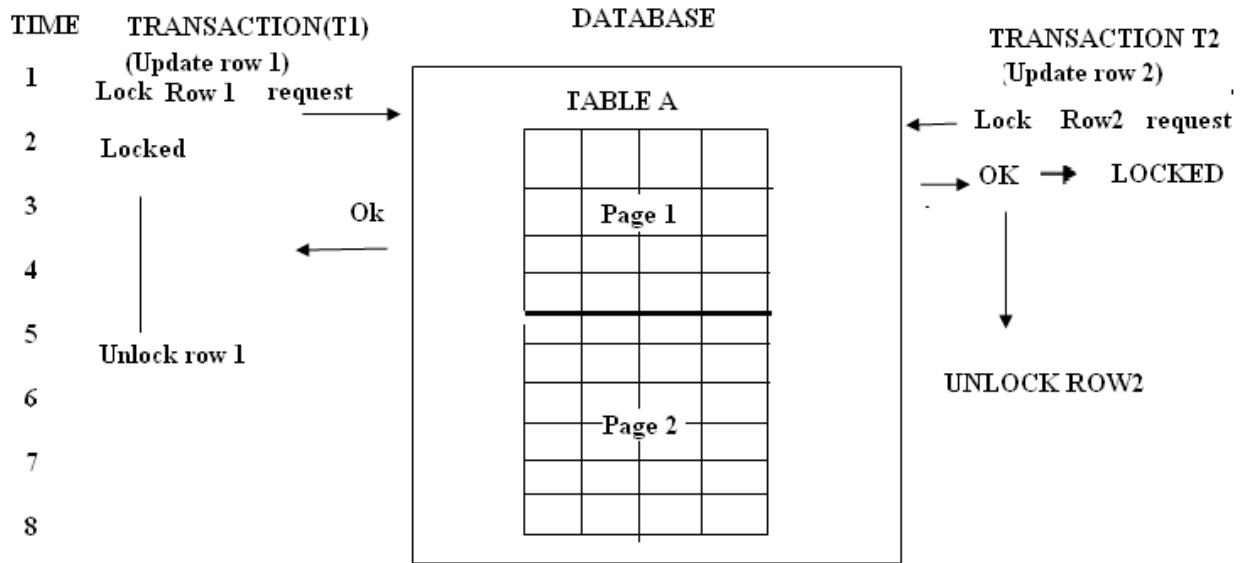
The following shows transactions access the same table while locking different pages. If T₂ requires the use of a row located on a page that is locked by T₁, T₂ must wait until the page is unlocked by T₁.



Row level

The DBMS allows concurrent transactions to access different rows of a same table even when the row are located on the same page. Although the row level locking approach the improves the availability of the rows but its management requires high overhead.

The following shows the row level lock.



In the above fig. the both transactions execute concurrently, even when the requested rows are on the same page. T2 must wait only if it requests the same row as T1.

Field Level

The field level lock allows concurrent transactions to access the same row but different fields. The field level locking gives most flexible multiuser data access but it requires an extremely high level overheads.

LOCK TYPES:

The DBMS may use different lock types.

1. Binary lock
2. Shared/ Exclusive lock.

Binary Lock: A binary lock has two states locked (1), unlocked (0). Every transaction requires a lock and unlock operations for each data item that is accessed such operations automatically managed by DBMS.

An Example of Binary Lock for Lost Updates Problem

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	LOCK PRODUCT	
2	T1	READ PROD_QOH	35
3	T1	PROD_QOH=35+100	
4	T1	WRITE PROD_QOH	135

5	T1	UNLOCK PRODUCT	
6	T2	LOCK PRODUCT	
7	T2	READ PROD_QOH	135
8	T2	PROD_QOH=135-30	
9	T2	WRITE PROD_QOH	105
10	T2	UNLOCK PRODUCT	

Shared or Exclusive lock

1. A shared lock is issued when a transactions wants to read data item from the database and no exclusive lock is held on that data item.
2. An Exclusive lock is issued when a transaction wants to update a data item and no locks are held on that data item by any other transaction.

The following table shows conflict when at least one of a transaction is a write operation.

	TRANSACTION		RESULT
	T1	T2	
OPERATION	READ	READ	No Conflict
	READ	WRITE	Conflict
	WRITE	READ	Conflict
	WRITE	WRITE	Conflict

For example if transaction **T₁** has a **shared lock** on a data item x and transaction T₂ wants to read data item x, **T₂ may also obtain a shared lock** on data item x.

For example if a **shared or exclusive lock is already held on data item x** by transaction T₁, **exclusive lock cannot be granted to transaction T₂** and T₂ must wait until T₁ completes. This condition is known as mutual exclusive rule.

Two Phase locking

The two phases are

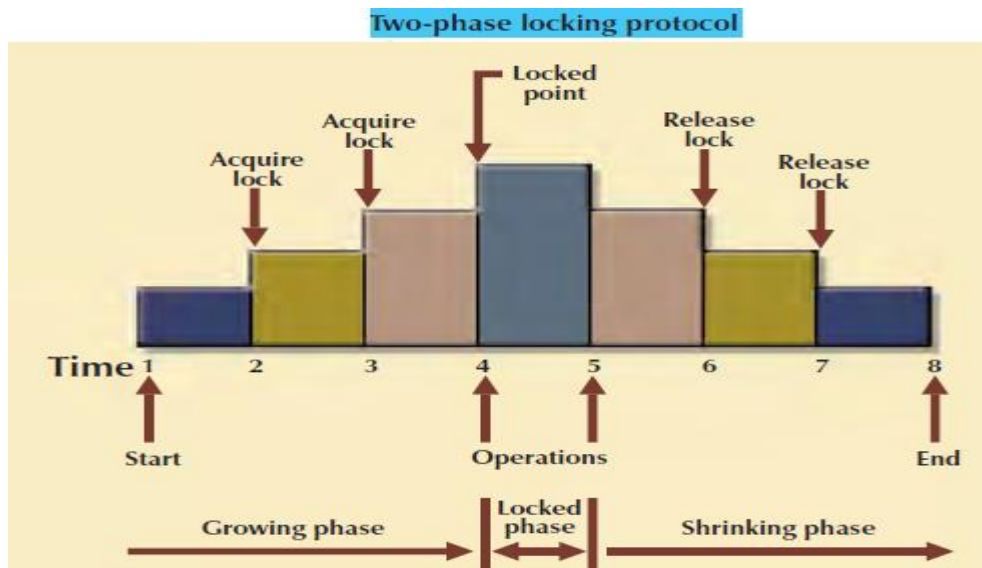
- i. Growing Phase
- ii. Shrinking Phase.

Growing Phase:

In growing phase the transaction acquires all required locks on data item. Once all locks have been acquired the transaction is in locked point and performs the operation.

Shrinking Phase:

When the transaction completed the operations then starts the shrinking phase. In shrinking phase the transaction releases all locks and cannot obtain any new locks.



The transaction acquires all the locks it needs. When the locked point is reached then performs the operation, when the transaction is completed it releases all the locks.

DEAD LOCK :


A dead lock occurs when two users have a lock each on separate resources. They want to acquire a lock on each other resources.

In this situation, the first user has to wait until the second user releases the lock and alternatively the second user also waits till the first user releases the lock. At this stage both the users are in a waiting state. They cannot proceed with their work.

The following table shows how a deadlock condition occurs.

How a Deadlock Condition Is Created

TIME	TRANSACTION	REPLY	LOCK STATUS	
0			Data X	Data Y
1	T1:LOCK(X)	OK	Unlocked	Unlocked
2	T2: LOCK(Y)	OK	Locked	Unlocked
3	T1:LOCK(Y)	WAIT	Locked	Locked
4	T2:LOCK(X)	WAIT	Locked	Locked
5	T1:LOCK(Y)	WAIT	Locked	Locked
6	T2:LOCK(X)	WAIT	Locked	Locked
7	T1:LOCK(Y)	WAIT	Locked	Locked
8	T2:LOCK(X)	WAIT	Locked	Locked
9	T1:LOCK(Y)	WAIT	Locked	Locked
...
...
...
...



Dead locks are possible only when one of the transactions wants to obtain an exclusive lock on a data item, no dead lock item can exists among shared locks.

The three basic techniques to control dead locks are.

1. Dead Lock Prevention.
2. Dead Lock Detection
3. Dead Lock Avoidance.

Dead Lock Prevention

A transaction requesting a new lock is aborted when there is possibility that a dead lock can occurs. If a transaction is aborted, all changes made by the transaction are rolled back and all locks obtained by the transaction are released.

Dead Lock Detection

The DBMS periodically test the database for dead locks. If a dead lock is found one of the transactions is rolled back and other transaction continues.

Dead Lock Avoidance

The transaction must obtain all the locks it needs before it can be executed.

The choice of the best deadlock control method to use depends on the database environment.

For example, if the Probability of deadlocks is low, deadlock detection is recommended. However, if the probability of deadlocks is high, deadlock prevention is recommended.

CONCURRENCY CONTROL WITH TIME STAMP METHODS

The time stamping approach to scheduling concurrent transactions assigns unique time stamp to each transaction. Based on the time stamp the transaction will be executed. All database operations within the same transaction must have the same time stamp. The DBMS executes conflicting operations in the time stamp order.

Two schemes used to time stamp to decide which transaction is rolled back and which transaction is continues , The schemes are wait/die and wound/waint

Wait/Die and Wound/Wait Scheme

Assume that we have two conflicting transactions T_1 and T_2 . T_1 has a time stamp 1154 and T_2 has a time stamp 1956. T_1 is older transaction and T_2 is younger transaction.

Wait/Die and Wound/Wait Concurrency Control Schemes

TRANSACTION REQUESTING LOCK	TRANSACTION OWNING LOCK	WAIT/DIE SCHEME	WOUND/WAIT SCHEME
T1 (11548789)	T2 (19562545)	<ul style="list-style-type: none"> T1 waits until T2 is completed and T2 releases its locks. 	<ul style="list-style-type: none"> T1 preempts (rolls back) T2. T2 is rescheduled using the same time stamp.
T2 (19562545)	T1 (11548789)	<ul style="list-style-type: none"> T2 dies (rolls back). T2 is rescheduled using the same time stamp. 	<ul style="list-style-type: none"> T2 waits until T1 is completed and T1 releases its locks.

Wait or Die scheme

1. If a transaction requesting a lock is the older of two transactions, the older transaction will wait until the younger transaction is completed and locks are released.
2. If a transaction requesting a lock is the younger of two transactions, the younger transaction will die (rollback) and younger transaction is rescheduled using the same time stamp.

Wound or Wait Scheme

1. If a transaction requesting a lock is the older of two transactions T_1 wounds T_2 (rollback), T_2 is rescheduled. The younger transaction is rescheduled using the same time stamp.
2. If a transaction requesting a lock is the younger of two transactions, the younger transaction will wait until the older transaction is completed.

DATABASE RECOVERY MANAGEMENT

Database recovery restores a database from a inconsistent state to consistent state. Any transaction operations cannot be completed, the transaction must be

rollback and any changes to the database must be rolled back. Recovery techniques applied to the database after some types of critical events occurred to the system.

Example of Critical Events

1. Hardware or Software failure

Hardware failures are the hard disk failure, a bad capacitor on the mother board. The software failures are the application program or the operating systems error that cause data to be deleted or lost.

Human Caused Incident

This type of events can be categorized as,

- Unintentional
- Intentional

An Unintentional Failure: Under this category, the humans are deleting the wrong rows from a table pressing the keys or shutdown the server by accidentally.

An Intentional Failure: Under this category, the unauthorized users accessing the database to perform operations on the database and virus attackers on the database to damage the data on the company.

Natural Incidents

Under this category earthquakes, floods and power failures. The critical events can render the database in an inconsistent state. The various techniques are used to recovery the database from an inconsistent state to consistent state.

Transaction Recovery

The following are the four important concepts that affect the recovery process.

1. The Write ahead protocol

The transaction logs are always written before any database data are updated.

2. Redundant Transaction Logs

We are maintaining several copies of transaction log in different storage devices.

3. Database Buffer

Database buffers are temporary storage areas in primary memory used to speed up disk operations.

4. Database Check Points

Data base check points are operations in which the DBMS writes all of its updated buffers to disk. Check point operations also registered in the transaction log. Check points are automatically scheduled by the DBMS several times per hour.

Transaction recovery procedure uses differed write and write through techniques.

Differed Write

In this technique the transaction operations do not immediately update the database, only the transaction log is updated. The database is updated only after the transaction reaches commit, using information from the transaction log, if the transaction aborts before it reaches commit, no changes made in the database by the transaction because the database was never updated.

The recovery process for all started and committed transaction follows these steps.

1. For a transaction that started and was committed before the last check point. Nothing needs to be done because the data already saved.
2. For a transaction that performed a commit operation after the last check point, the DBMS uses the transaction log records to redo the transaction and to update the database, using the after values in the transaction log.
3. For a transaction that has a roll back operation after the last check point, nothing needs to be done because the database was never updated.

Write through technique

In this technique the database is immediately updated by the transaction operation even before the aborts or before it reaches commit point a undo operation needs to restore the database to a consistency. The recovery process follows these steps.

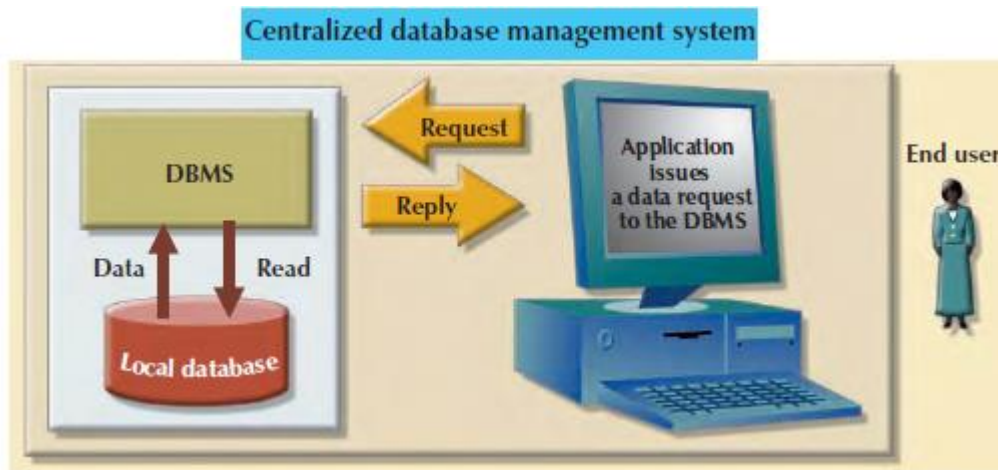
1. For a transaction that started and was committed before the last check point. Nothing needs to be done because the data are already saved.
2. For a transaction that was committed after the last check point, the DBMS uses the transaction log records to redo the transaction, using the after values in the transaction log.
3. For any transaction that had a roll back operation after the log check point the DBMS uses the transaction log to undo the operations using the before values in the transaction log.

DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

Centralized Database System:

A Centralized database is a database which is located and maintained in one location, unlike a distributed database. One main advantage of centralized database is that all data is located in one place and the disadvantage is that bottlenecks may occur.

The use of a centralized database required that corporate data be stored in a single central site, usually a mainframe computer. Data access was provided through dumb terminals.



The centralized approach worked well to fill the structural information needs of corporations. The centralized database management system fell short when faster response and quick access of information

What was needed from centralized database management system was quick access, unstructured information and using adhoc queries to generate on the spot information.

Because of above problems the technology changed , that effect the database development and design.

A Distributed Database Management System (DDBMS) consists of a single logical database that is split into a number of fragments. Each fragment is stored on one or more computers under the control of a separate DBMS, with the computers connected by a communications network. Each site is capable of independently processing user requests that require access to local data (that is, each site has some degree of local autonomy) and is also capable of processing data stored on other computers in the network.

Distributed DBMS Advantages and Disadvantages

ADVANTAGES DISADVANTAGES

- Data are located near the greatest demand site.
- **Faster data access.** End users often work with only a locally stored subset of the company's data.
- **Faster data processing.** A distributed database system spreads out the systems workload by processing data at several sites.
- **Growth facilitation.** New sites can be added to the network without affecting the operations of other sites.
- **Improved communications.** Because local sites are smaller and located closer to customers, local sites foster better communication among departments and between customers and company staff.
- **Reduced operating costs.** It is more cost-effective to add workstations to a network than to update a mainframe system. Development work is done more cheaply and more quickly on low-cost PCs than the mainframes.
- **User-friendly interface.** PCs and workstations are usually equipped with an easy-to-use graphical user interface (GUI). The GUI simplifies training and use for end users.
- **Less danger of a single-point failure.** When one of the computers fails, the workload is picked up by other workstations. Data are also distributed at multiple sites.
- **Processor independence.** The end user is able to access any available copy of the data, and an end user's request is processed by any processor at the data location.

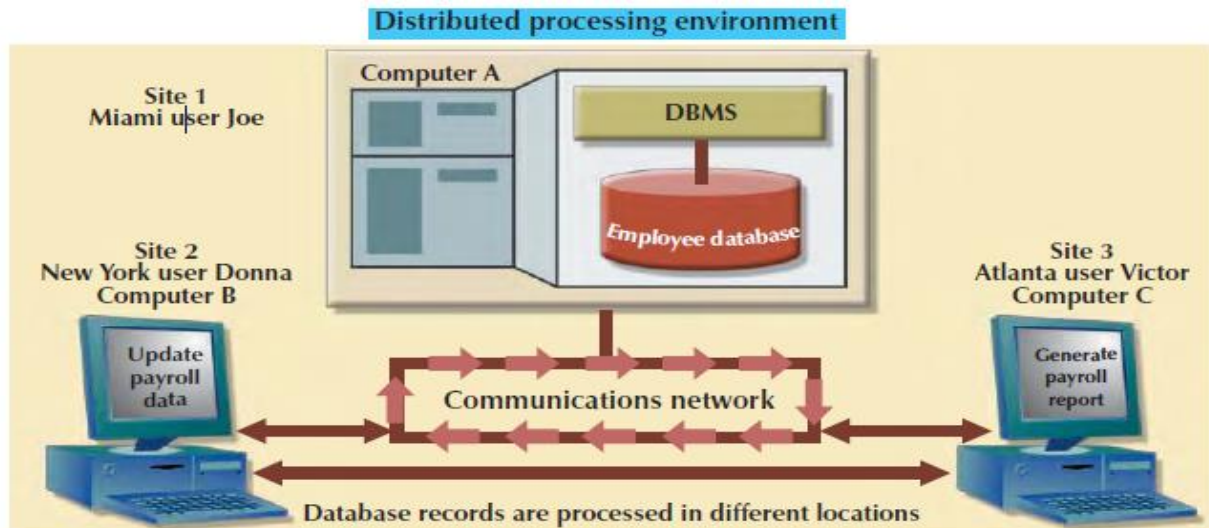
Disadvantages:

- **Complexity of management and control.** Applications must recognize data location, and they must be able to stitch together data from various sites.
- **Technological difficulty.** Data integrity, transaction management, concurrency control, security, backup, recovery, query optimization, access path selection, and so on, must all be addressed and resolved.
- **Security.** The probability of security lapses increases when data are located at multiple sites. The responsibility of data management will be shared by different people at several sites.
- **Lack of standards.** There are no standard communication protocols at the database level.
- **Increased storage and infrastructure requirements.** Multiple copies of data are required at different sites, thus requiring additional disk storage space.
- **Increased training cost.** Training costs are generally higher in a distributed model than the centralized model

- **Costs.** Distributed databases require duplicated infrastructure to operate (physical location, environment, personnel, software, licensing, etc.)

Distributed processing:

In **distributed processing**, a database's logical processing is shared among two or more physically independent sites that are connected through a network.

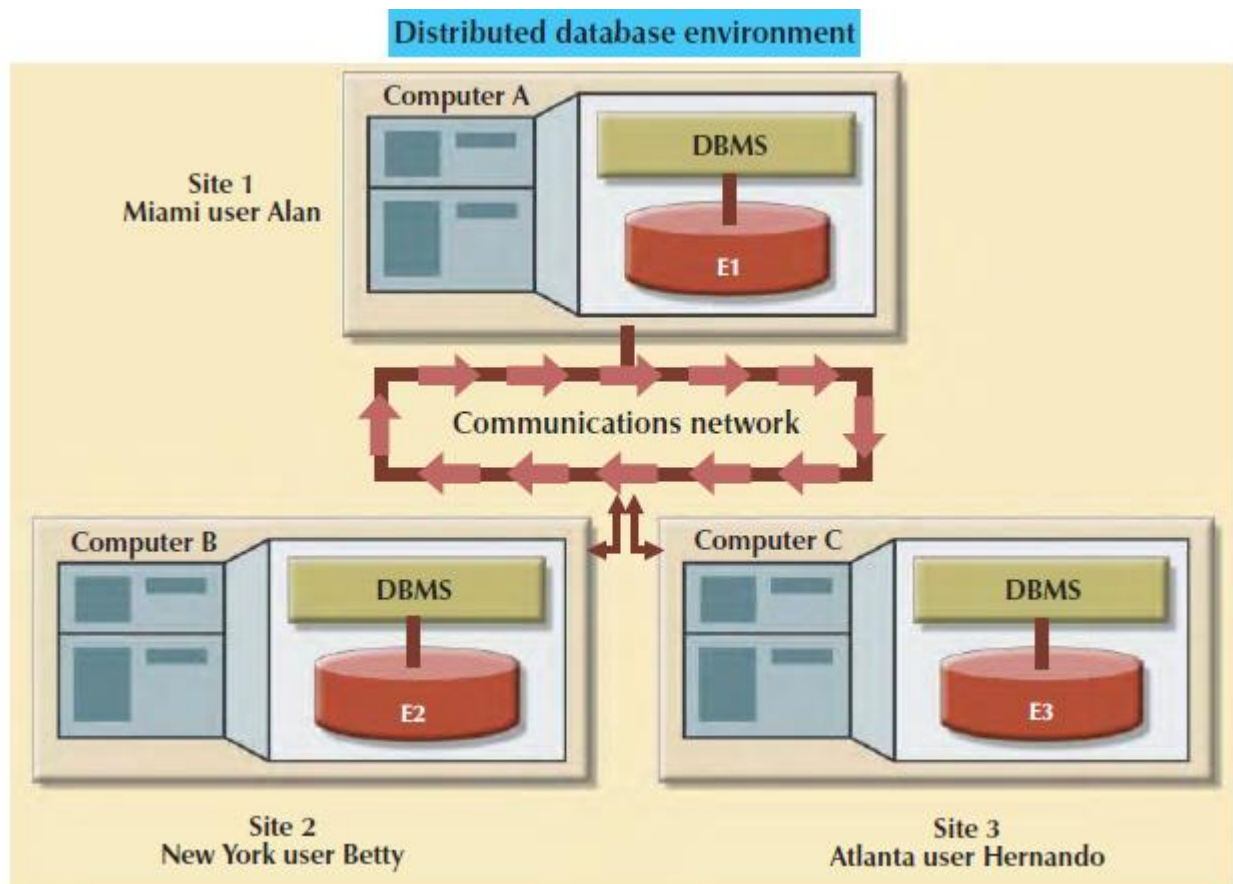


Although the database resides at only one site, each site can access the data and update the database. The database is located on Computer A, a network computer known as the *database server*.

Distributed Database:

A Distributed database stores a logically related data base in two or more sites. The sites are connected via a communication network.

In distributed database system a database is composed of of several parts known as database fragments. The database fragments are located at different sites. An example of distributed database environment is shown below.



In the above diagram the database is divided into three fragments (E1,E2,E3) located at different sites. When you compare the distributed database the following points are observed.

1. Distributed processing does not require a distributed database, but a distributed database requires distributed processing (each database fragment is managed by its own local database process).
2. Both distributed processing and distributed databases require a network to connect all components.

Characteristics of distributed database management system:

The Characteristics of distributed database management system are

1. **Application interface:** To interact with the end user, application programs, and other DBMSs within the distributed database.
2. **Validation:** To analyze data requests for syntax correctness.
3. **Transformation:** To decompose complex requests into atomic data request components
4. **Mapping:** To determine the data location of local and remote fragments.
5. **I/O interface:** To read or write data from or to permanent local storage.

6. **Security** : To provide data privacy at both local and remote databases.
7. **Backup and recovery**: To ensure the availability and recoverability of the database in case of a failure.
8. **Concurrency control**: To manage simultaneous data access and to ensure data consistency across database fragments in the DDBMS.
9. **Transaction management**: To ensure that the data moves from one consistent state to another. This activity includes the synchronization (bringing together) of local and remote transactions as well as transactions across multiple distributed segments.

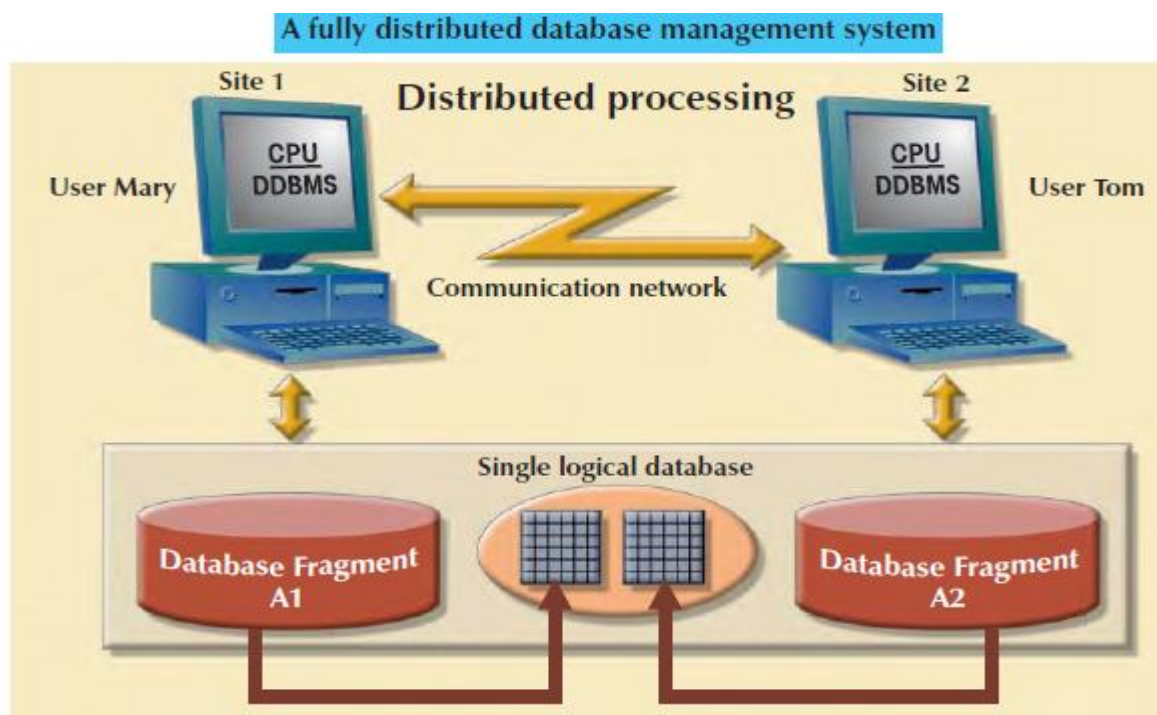
DDBMS COMPONENTS:

The components of DDBMS are

Computer workstations: It form the network system. The distributed database system must be independent of the computer system hardware.

Network hardware and software: the network hardware and software components that resides in each workstation. The network components allow all sites to interact and exchange data.

Communications media: The communication media that carry the data from one node to another.

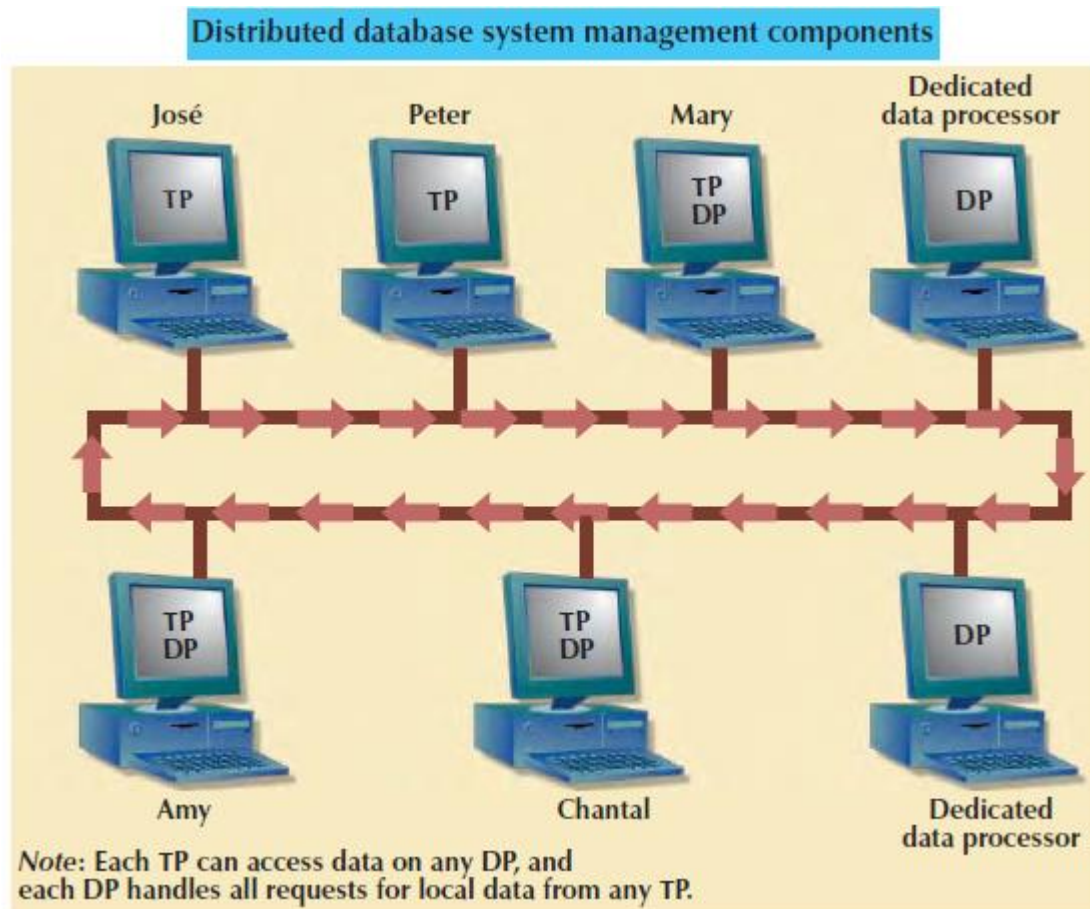


Transaction processor (TP):

The **transaction processor (TP)**, which is the software component found in each computer or device that requests data. The transaction processor receives and processes the application's data requests (remote and local). The TP is also known as the **application processor (AP)** or the **transaction manager (TM)**.

Data processor (DP):

The **data processor (DP)**, which is the software component residing on each computer or device that stores and retrieves data located at the site. The DP is also known as the **data manager (DM)**.



LEVELS OF DATA AND PROCESS DISTRIBUTION:

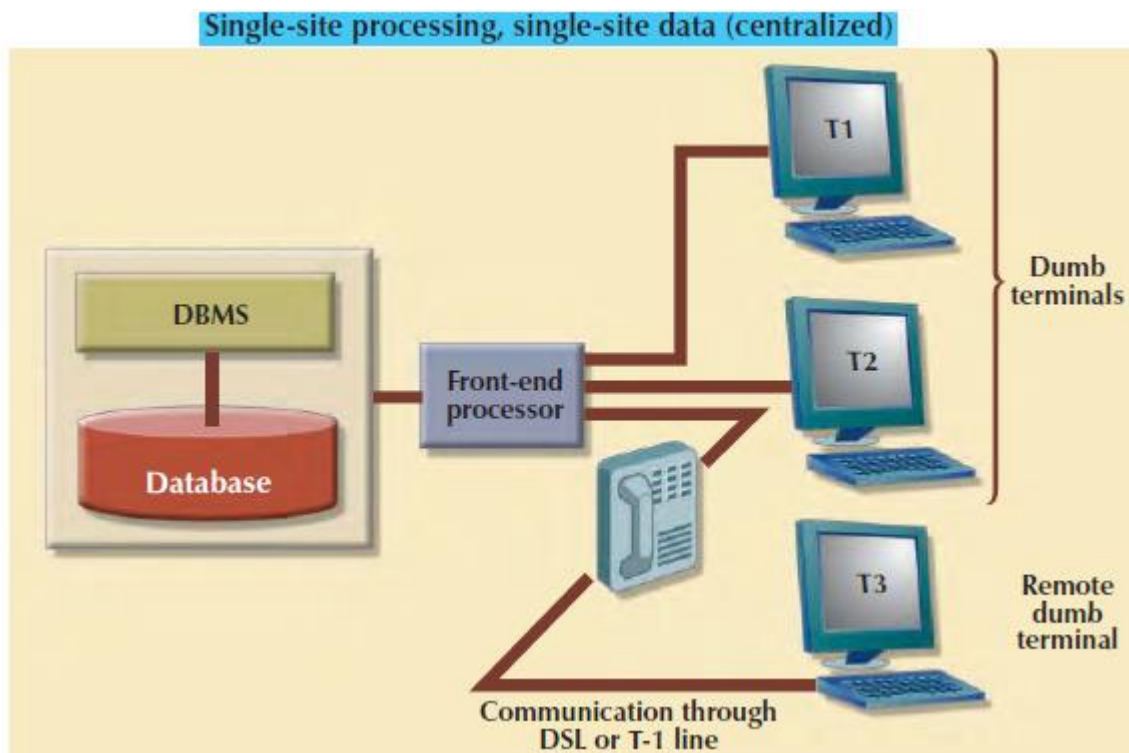
The database system can be classified on the basis of process distribution and data distribution.

1. Single Site processing , Single site data (SPSD)
2. Multisite processing, single site data (MPSD)
3. Multisite processing , Multisite data (MPMD)

Single Site processing , Single site data (SPSD)

In SPSD all processing is done on a single host computer (mainframe computer) and all data are stored on the host computer's hard disk.

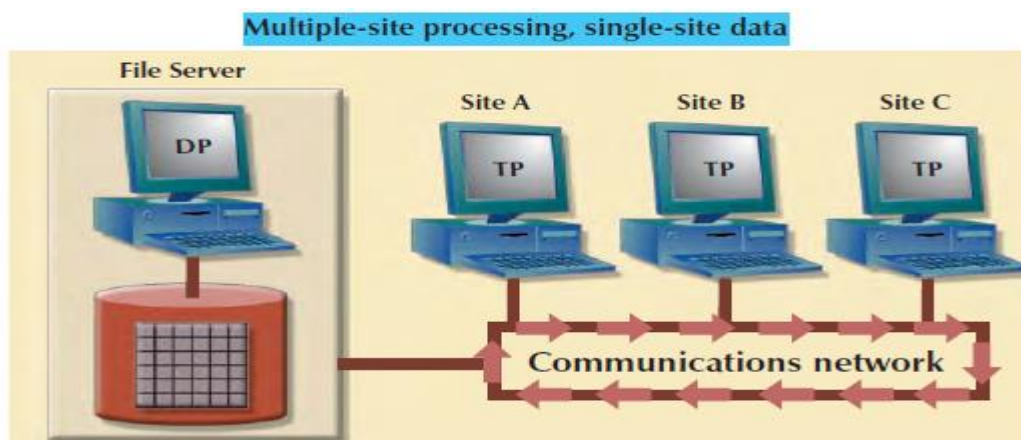
Processing cannot be done on the end user's side of the system.



The transaction processor and Data processor are embedded with in the DBMS located on a single computer.

Multiple-Site Processing, Single-Site Data (MPSD)

Under the **multiple-site processing, single-site data (MPSD)** scenario, multiple processes run on different computers sharing a single data repository. The MPSD scenario requires a network file server.



All records and the files locking activities are done at the work station.

All data selection, search, and update functions take place at the workstation, thus requiring that entire files travel through the network for processing at the workstation. Such a requirement increases network traffic, slows response time, and increases communication costs.

Multiple-Site Processing, Multiple-Site Data (MPMD):

The **multiple-site processing, multiple-site data (MPMD)** scenario supports multiple transaction processing and multiple data processing at multiple sites. The DDBMSs are classified as either homogeneous or heterogeneous.

Homogeneous DDBMSs: Homogeneous DDBMS supports same DBMS will be running on different server platforms.

Heterogeneous DDBMSs: Heterogeneous DDBMS supports different DBMS will be running on different server platforms.

DISTRIBUTED DATABASE TRANSPARENCY FEATURES:

The DDBMS transparency features are:

Distribution transparency, which allows a distributed database to be treated as a single logical database. If a DDBMS exhibits distribution transparency, the user does not need to know:

- a. About the data are partitioned (meaning the table's rows and columns are split vertically or horizontally and stored among multiple sites.)
- b. That the data can be replicated at several sites.
- c. The data location.

Transaction transparency, which allows a transaction to update data at more than one network site. Transaction transparency ensures that the transaction will be either entirely completed or aborted, thus maintaining database integrity.

Failure transparency, which ensures that the system will continue to operate in the event of a node failure.

Performance transparency: The system will not suffer any performance degradation due to its use on a network or due to the network's platform differences.

Heterogeneity transparency, which allows the integration of several different local DBMSs under a common, or global, schema. The DDBMS is responsible for translating the data requests from the global schema to the local DBMS schema.

DISTRIBUTION TRANSPARENCY:

Three levels of distribution transparency are recognized:

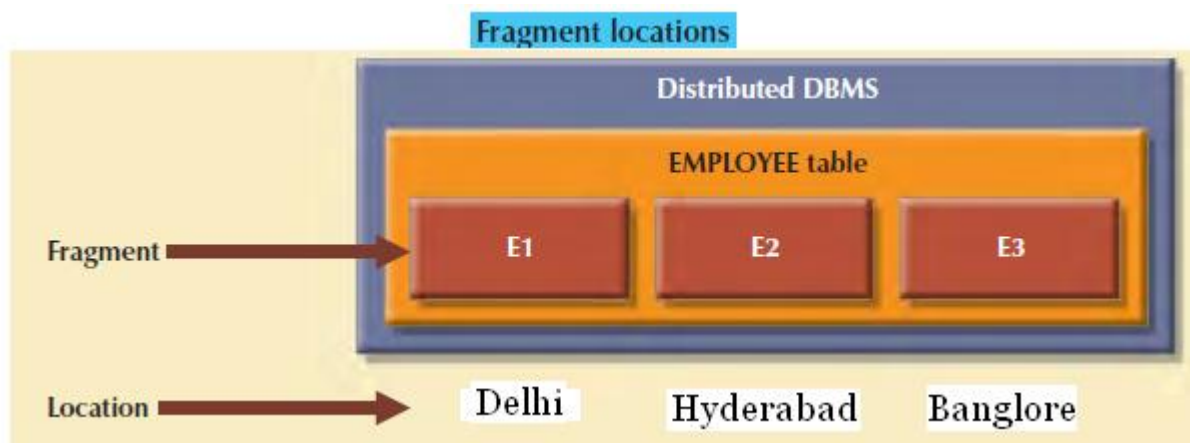
Fragmentation transparency: It is the highest level of transparency. The end user or programmer does not need to know the fragment names and fragment locations to access data.

Location transparency: This is the middle level of transparency. The end user must specify fragment name and does not need to specify the location name.

Local mapping transparency: This is the lowest level of transparency. The end user or programmer must specify both the fragment names and their locations name.

For example the employee data are distributed over three different locations – Delhi, Hyderabad, Bangalore.

The Delhi employees data are stored in fragment E1, Hyderabad employees data are stored in fragment E2, Bangalore employees data are stored in fragment E3 is shown below.



Suppose the end user wants to list all the employees with a date of birth before Jan 1,1980. The end user written this query in the fragmentation transparency as shown below.

```
Select * from employees where DOB < '01-jan-1980';
```

The end user written this Query in the location Transparency as shown below.

```
Select * from E1 where DOB < '01-jan-1980'
```

```
Union
```

```
Select * from E2 where DOB < '01-jan-1980'
```

```
Union
```

```
Select * from E3 where DOB < '01-jan-1980';
```

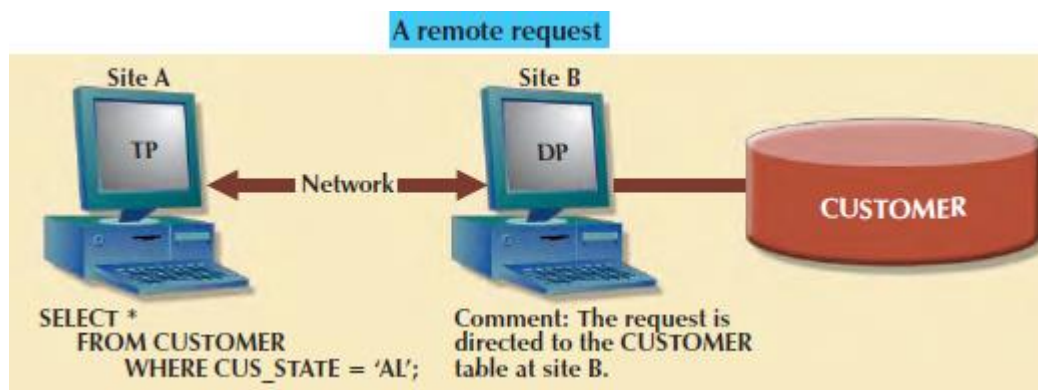
The end user written this Query in the local Transparency as shown below.

```
Select * from E1 Delhi where DOB < '01-jan-1980'
Union
Select * from E2 Hyderabad where DOB < '01-jan-1980'
Union
Select * from E3 Bangalore where DOB < '01-jan-1980';
```

TRANSACTION TRANSPARENCY

To understand how the transactions are managed in the DDBMS, we should know the basic concepts of remote request, remote transaction, Distributed request and Distributed transaction.

Remote Request: A Remote request is shown below



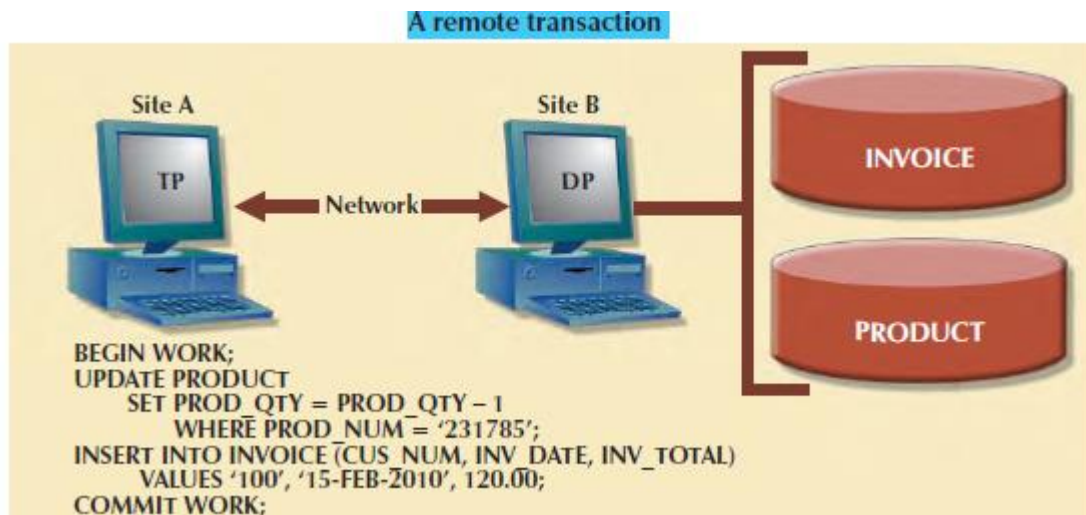
Select * from customer where cus_state='AL';

The request is directed to customer table at site B.

The single SQL statement access the data that are to be processed by a single Remote Database processor.

Remote Transaction:

Similarly, a **remote transaction**, composed of several requests, accesses data at a single remote site. A remote transaction is shown below.



```
BEGIN WORK;
```

```
UPDATE PRODUCT SET PROD_QTY = PROD_QTY - 1
```

```
WHERE PROD_NUM = '231785';
```

```
INSERT INTO INVOICE (CUS_NUM, INV_DATE, INV_TOTAL)
```

```
VALUES '100', '15-FEB-2010', 120.00;
```

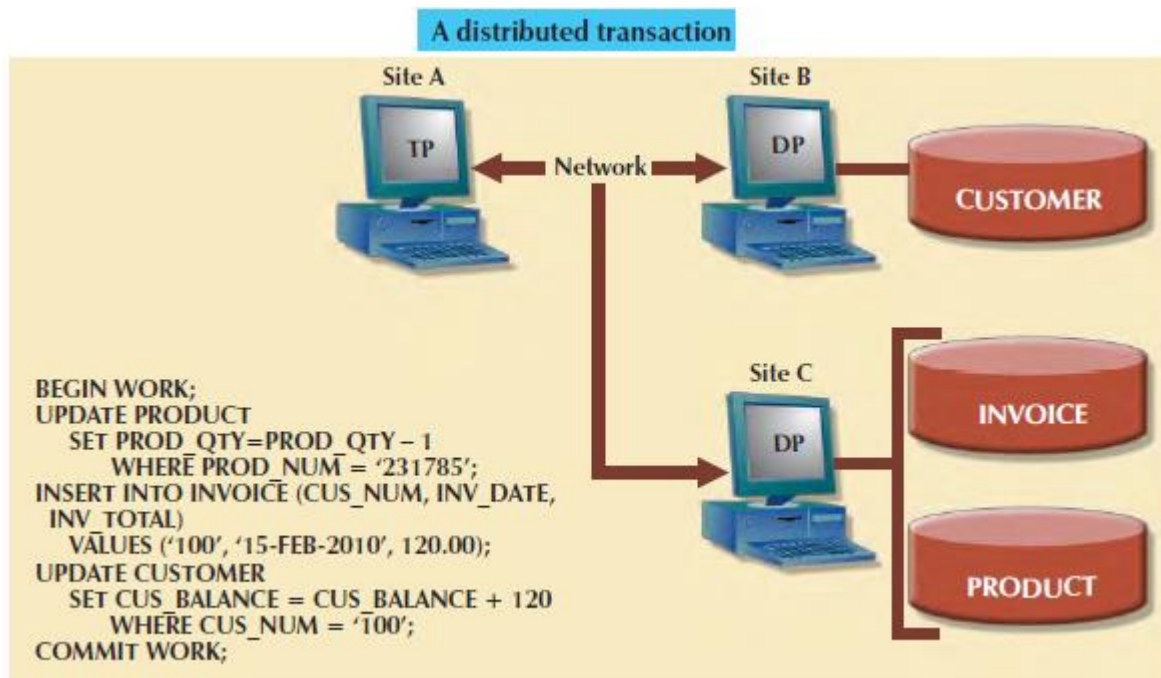
```
COMMIT WORK;
```

As we examine the above diag. we observe the following points.

1. The transaction updates the product and invoice table (located at site B).
2. The remote transaction is send to and executed at site B.

Distributed Transaction:

A distributed transaction allows a transaction to reference several different local or remote DP sites.



```

BEGIN WORK;
UPDATE PRODUCT
SET PROD_QTY=PROD_QTY - 1 WHERE PROD_NUM = '231785';
INSERT INTO INVOICE (CUS_NUM, INV_DATE, INV_TOTAL) VALUES ('100', '15-FEB-
2010', 120.00);
UPDATE CUSTOMER SET CUS_BALANCE = CUS_BALANCE + 120
WHERE CUS_NUM = '100';
COMMIT WORK;

```

As we observe the above diag. the following features are

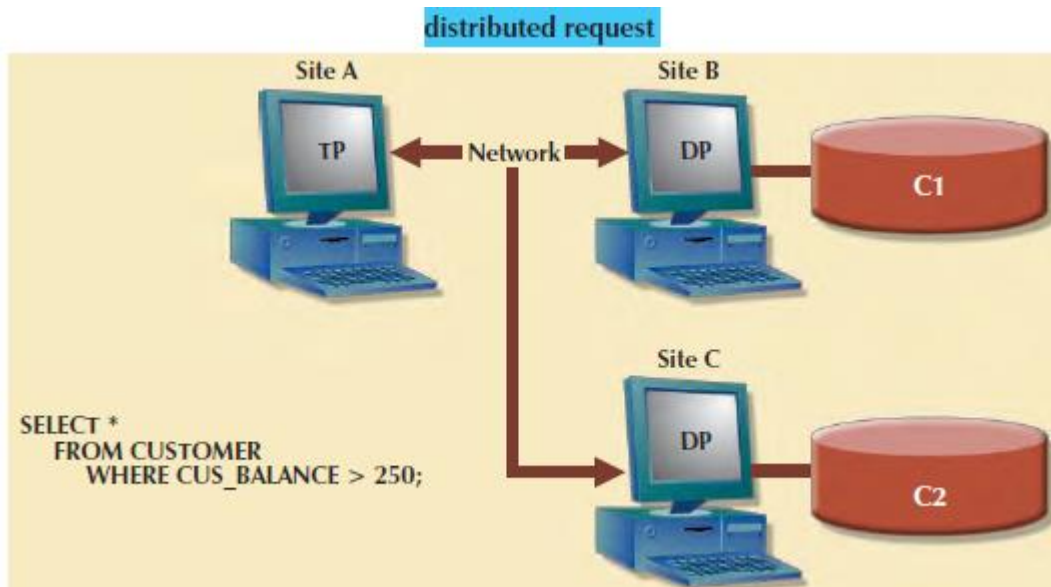
1. The transaction references two remote sites (B and C).
2. The first two requests (UPDATE PRODUCT and INSERT INTO INVOICE) are processed by the DP at the remote site C, and the last request (UPDATE CUSTOMER) is processed by the DP at the remote site B.
3. Each request can access only one remote site at a time.

The third characteristic may create problems. For example, suppose the table PRODUCT is divided into two fragments, PROD1 and PROD2, located at sites B and C, respectively, the distributed transaction cannot access the data from more than one site. Therefore the DDBMS must support a distributed request.

Distributed request:

The Distributed request allows a single request to reference a partitioned table in several sites.

For ex. Customer table is divided into two fragments E1 and E2 located at site B and Site C are shown below.



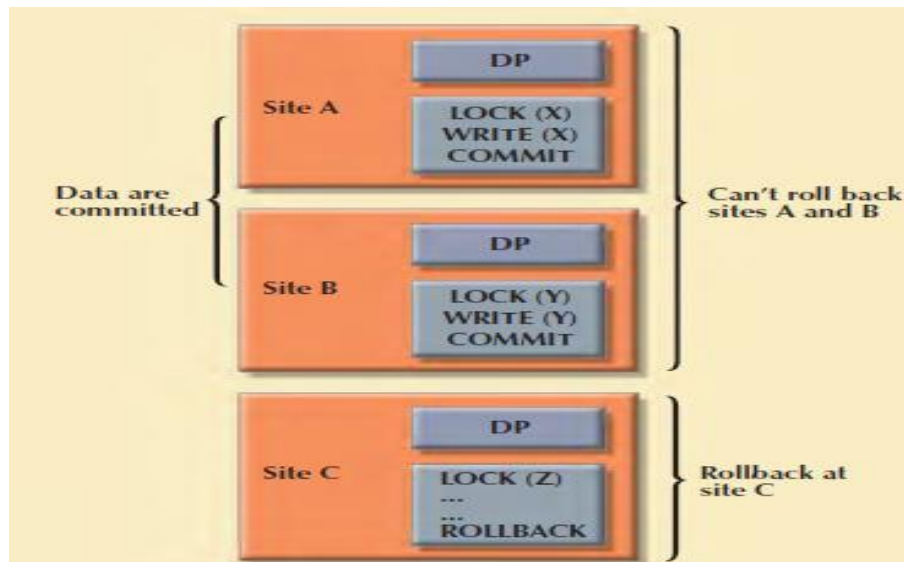
`SELECT * FROM CUSTOMER WHERE CUS_BALANCE > 250;`

DISTRIBUTED CONCURRENCY CONTROL:

Concurrency control is very important in the distributed database environment because multisite, multiple-process operations are create data inconsistencies and deadlocked transactions.

Suppose that each transaction operation was committed by each local DP, but one of the DPs could not commit the transaction. Then the transaction is 'Aborted'. Otherwise the transaction(s) would yield an inconsistent database.

The solution for the problem is a *two-phase commit protocol*.



Two phase protocol:

In distributed database a transaction access data at several sites. A final commit must not be issued until all sites have committed their parts of the transaction.

The two phase commit protocol guaranties that if a portion of a transaction operations cannot be committed all changes made at the others sites by the transaction will be aborted.

Each DP maintains its own transaction log. The transaction log for each DP written before the database fragment is updated.

The two phase commit protocols requires a DO-UNDO-REDO protocol and a write a head protocol. The DO-UNDO-REDO protocols defines three types of Operations.

- Performs the operations and before and after values in the transaction log.
- Undo reverses an operation, using the log written by DO operation.
- REDO reduces the operation using the Transaction Log written by the DO portion of sequences.

The Two-Phase commit protocol defines the operations between two types of nodes: the **coordinator** and one or more **subordinates**, or *cohorts*. The participating nodes agree on a coordinator.

The protocol is implemented in two phases:

Phase 1: Preparation

The coordinator sends a PREPARE TO COMMIT message to all subordinates.

- The sub coordinator receive the message; write the transaction log, using the write-ahead protocol; and send an acknowledgment (YES/PREPARED TO COMMIT or NO/NOT PREPARED) message to the coordinator.

2. If all nodes are PREPARED TO COMMIT, the transaction goes to Phase 2. If one or more nodes reply NO or NOT PREPARED, the coordinator broadcasts an ABORT message to all subordinates.

Phase 2: The Final COMMIT

1. The coordinator broadcasts a COMMIT message to all subordinates and waits for the replies.
2. Each subordinate receives the COMMIT message, and then updates the database using the DO protocol.
3. The subordinates reply with a COMMITTED or NOT COMMITTED message to the coordinator.

If one or more subordinates did not commit, the coordinator sends an ABORT message, thereby forcing them to UNDO all changes.

Note: The objective of the two-phase commit is to ensure that each node commits its part of the transaction; otherwise, the transaction is aborted.

PERFORMANCE TRANSPARENCY AND QUERY OPTIMIZATION:

In centralized database all data resides at a single site. The DBMS evaluate data request and find the most efficient way to access the data.

In DDBMS the database is divided into several fragments. In DDBMS Query evaluation is more complicated, because the DDBMS must decide which fragment of data is access. In addition the data may also repeated several times.

The DDBMS uses query optimization technique to deal with such problems.

The objective of the query optimization is to minimize the total execution cost of the request.

Access time (I/O) cost: It involved in accessing the physical data stored on disk.

Communication cost: It associated with the transmission of data among nodes in distributed database systems.

CPU time cost: It associated with the processing overhead of managing distributed transactions.

The most important characteristics of query optimization in distributed database systems is that it must provide distribution transparency as well as *replica* transparency.

Replica transparency refers to the DDBMS's ability to hide the existence of multiple copies of data from the user.

Query optimizations are based on two principles:

- a) The selection of the optimum execution order.
- b) The selection of sites to be accessed to minimize communication costs.

Query optimization algorithm can be evaluated the request on the basis of its *operation mode* or the *timing of its optimization*.

Operation modes can be classified as manual or automatic.

Automatic query optimization automatically scheduled by the transaction

Manual query optimization scheduled by the end user or programmer.

Timings of its Optimization: Timing Optimization mode can be classified as static or Dynamic

Static query optimization: When the program is submitted to the DBMS for compilation, it creates the necessary plan to access the database. When the program is executed, the DBMS uses that plan to access the database.

Dynamic query optimization. Database access plan defined when the program is executed.

Finally, query optimization techniques can be classified according to the type of information that is used to optimize the query.

A **statistically based query optimization algorithm** uses statistical information about the database.

The statistics provide information about database characteristics such as size, number of records, average access time, number of requests serviced, and number of users with access rights.

The statistical information is managed by the DDBMS and is generated in one of two different modes: dynamic or manual.

In the **dynamic statistical generation mode**, the DDBMS automatically evaluates and updates the statistics after each access.

In the **manual statistical generation mode**, the statistics must be updated periodically.

A **rule-based query optimization algorithm** is based on a set of user-defined rules to determine the best query access strategy.

DISTRIBUTED DATABASE DESIGN:

Fragmentation:

Data fragmentation allows you to break a single object into two or more segments, or fragments. The object might be a user's database, a system database, or a table. Each fragment can be stored at any site over a computer network.

Horizontal fragmentation partitions the rows of a global fragment into subsets. A fragment r_1 is a *selection* on the global fragment r using a predicate P_i , its *qualification*. The reconstruction of r is obtained by taking the union of all fragments.

Vertical fragmentation subdivides the attributes of the global fragment into groups. The simplest form of vertical fragmentation is decomposition. A unique *row-id* may be included in each fragment to guarantee that the reconstruction through a join operation is possible.

Mixed fragmentation is the result of the successive application of both fragmentation techniques.

Rules for Fragmentation

1. Fragments are formed by the select predicates associated with dominant database transactions. The predicates specify attribute values used in the conjunctive (AND) and disjunctive (OR) form of select commands, and rows (records) containing the same values form fragments.
2. Fragments must be disjoint and their union must become the whole fragment. Overlapping fragments are too difficult to analyze and implement.
3. The largest fragment is the whole table. The smallest table is a single record. Fragments should be designed to maintain a balance between these extremes.

C. J. DATE'S TWELVE COMMANDMENTS FOR DISTRIBUTED DATABASES

Rule-1.

Local Autonomy. Local data is locally owned and managed, even when it is accessible by a remote site. Security, integrity, and storage remain under control of the local system. Local users should not be hampered when their system is part of a distributed system.

Rule-2.

No Central Site. There must be no central point of failure or bottleneck. Therefore the following must be distributed: dictionary management, query processing, concurrency control, and recovery control.

Rule-3.

Continuous Operation. The system should not require a shutdown to add or remove a node from the network. User applications should not have to change when a new network is added, provided they do not need information from the added node.

Rule-4.

Location Independence (or Transparency). A common global user view of the database should be supported so that users need not know where the data is located.

This allows data to be moved for performance considerations or in response to storage constraints without affecting the user applications.

Rule-5.

Fragmentation Independence (or Transparency). This allows tables to be split among several sites, transparent to user applications. For example, we can store New York employee records at the New York site and Boston employees at the Boston site, but allow the user to refer to the separated data as EMPLOYEES, independent of their locations.

Rule-6.

Replication Independence (or Transparency). This allows several copies of a table (or portions thereof) to reside at different nodes. Query performance can be improved since applications can work with a local copy instead of a remote one. Update performance, however, may be degraded due to the additional copies. Availability can improve.

Rule-7.

Distributed Query Processing. No central site should perform optimization; but the submitting site, which receives the query from the user, should decide the overall strategy. Other participants perform optimization at their own levels.

Rule-8.

Distributed Transaction Processing. The system should process a transaction across multiple databases exactly as if all of the data were local. Each node should be capable of acting as a coordinator for distributed updates, and as a participant in other transactions. Concurrency control must occur at the local level (Rule 2), but there must also be cooperation between individual systems to ensure that a "global deadlock" does not occur.

Rule-9.

Hardware Independence. The concept of a single database system must be presented regardless of the underlying hardware used to implement the individual systems.

Rule-10.

Operating System Independence. The concept of a single database system must be presented regardless of the underlying operating systems used.

Rule-11.

Network Independence. The distributed system must be capable of communicating over a wide variety of networks, often different ones in the same configuration. Standard network protocols must be adhered to.

Rule-12.

DBMS Independence (Heterogeneity). The distributed system should be able to be made up of individual sites running different database management systems.