KBC North Maharashtra University, Jalgaon
T. Y. B. Sc. (CS and IT) (w.e.f. June -2020)
UG-CS 312 Database Management System

Semester V **Lectures: 60 Total Marks: 90**

1. Introduction of DBMS L 12: M 16

1.1. Overview, Definition
1.2. Types of DBMS
1.3. Describing & storing data (Data models (relational, hierarchical, network)),
1.4. Levels of abstraction, data independence,
1.5. Queries in DBMS (SQL: DDL, DML, DCL, TCL), Users of DBMS, Advantages of DBMS

2. Conceptual Design (E-R model) L 12 : M 16

2.1. Overview of DB design,
2.2. ER data model (entities, attributes, entity sets, relations, relationship sets) ,
2.3. Conceptual design using ER (entities VS attributes, Entity Vs relationship, Binary Vs ternary)

3. Relational data model L 12 : M 18

3.1. Relations (concepts, definition),
3.2. Conversion of ER to Relational model,
3.3. Integrity constraints (key, referential integrity, general constraints)
3.4 Codd's Rules, Functional Dependency, Data Normalization (1NF, 2NF, 3NF, BCNF)

4. Relational algebra L 08 : M

4.1. Preliminaries
4.2. Relational algebra (selection, projection, set operations, renaming, joins, division)

5. Database Implementations L-08 M:12

5.1 Database security,
5.2 Database integrity,
5.3 Transaction Concept,
5.4 Transaction State,
5.5 Transaction Properties (ACID)

6. Concurrency control, Backup &recovery:- L-08 M-12

6.1 Lock-Based protocol,
6.2 Timestamp-Based protocol,
6.3 Log base Recovery,
6.4 Shadow Paging,
6.5 Differed Updates.

1. Introduction to DBMS

1.1 Overview, Definition:

Information:

Information is a useable answer to a concrete question.

Data : The presentation of facts, information or concepts which are created in a computer readable form or are translated into such a form.

DBMS (Data Base Management System)

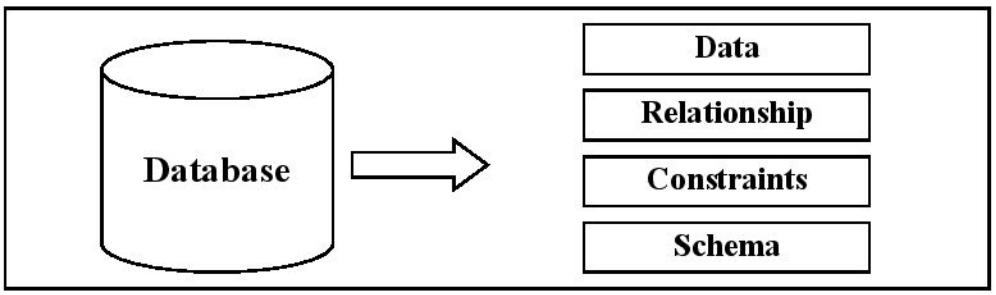
Which manages database is known as DBMS.

Data Base

A database consists of four elements.

1. Data
2. Relationship
3. Constraints
4. Schema

This is shown in following figure.



1. **Data**

   It is nothing but binary computer representation. It can exist in a Varity of forms i.e. number or text, stored in memory. That is it's single piece of information.

   The term data is often use to distinguish binary information (machine readable) from textual information (Human readable). In database management system, data files are the files which store the database information.

2. **Relationship**

   It represents correspondence between the various data elements.

3. **Constraints** (Rule)

   Constraints are predicates which define correct database state.

4. **Schema**

   It describes the organization of data and relationship within the database.

5. **Database** :A collection of different types of record

- **Database Management System :**

A database management system (DBMS) is collection of programs that Enables to **store, modify and extract** information from database.

DBMS contains information about a particular enterprise .Collection of interrelated data

Set of programs to access the data .An environment that is both *convenient* and *efficient* to use

Following are the Definitions

Data is a collection of facts, such as numbers, words, measurements, observations or just descriptions of things.
Database : Collection of Different types of records or A database is a data structure that stores organized information.
***For example**, a company database may include tables for products, employees, and financial records. Each of these tables would have different fields that are relevant to the information stored in the table.*
RECORD = Student Rollno : Numeric Name : Character Class : Character
Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database.

1.2 Types of DBMS

The main Four Types of Database Management System are:

- Hierarchical database
- Network database
- Relational database
- Object-Oriented database

Hierarchical DBMS

In a Hierarchical database, model data is organized in a tree-like structure. Data is Stored Hierarchically (top down or bottom up) format. Data is represented using a parent-child relationship. In Hierarchical DBMS parent may have many children, but children have only one parent.

Network Model

The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like as the orders/parts many-to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths.

Relational Model

Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL.

Object-Oriented Model

In Object-oriented Model data stored in the form of objects. The structure which is called classes which display data within it. It is one of the components of DBMS that defines a database as a collection of objects which stores both data members values and operations.

Data Model

Data Model gives us an idea that how the final system will look like after its complete implementation. It defines the data elements and the relationships between the data elements. Data Models are used to show how data is stored, connected, accessed and updated in the database management system. Here, we use a set of symbols and text to represent the information so that members of the organisation can communicate and understand it. Though there are many data models being used nowadays but the Relational model is the most widely used model.

The following are examples of database applications :

- Computerized library system.
- Automated teller machine.
- Airlines: reservations, schedules
- Universities: registration, grades
- Flight reservation system.
- Computerized parts inventory system.
- College Admission system
- Payroll system
- Banking: all transactions
- Sales: customers, products, purchases

➢ Database Management Systems Software :

These are called as **back end software**.

1. First generation database systems :

These are file based systems. They are structured databases, they are using files in the backend. Each table is stored as a separate file. There is no relation between files (tables).

DBase
Clipper
Foxbase
FoxPro

Visual FoxPro

2. Second generation database system :

Microsoft introduced MS Access as a database system to compete with other industry leading database systems. MS Access is still file based, but all tables in a database is merged into one file.

3. Third generation database systems

SQL Server
Oracle
Sybase

When you want to store or retrieve data, you will access the database server to do this job. You have to use a different application (like C#, VB.NET, C++) etc and communicate with the database server using some technology like ADO.NET.

Database servers understand a specific language called SQL.

List of DBMS software :

Oracle, DB2, Sybase Adaptive Server Enterprise, Ingres, Informix ,Microsoft Access ,Microsoft SQL Server, Microsoft Visual FoxPro, MySQL, PostgreSQL

1.3 Describing and Storing Data:

Data Models : The structure of database is called data model.

A collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.

Data model are divided into three different groups.

1) Physical data model.
2) Record based logical model.
3) Object based logical mode.

1. **Physical data model:-**

Physical data model are used to describe data at the lowest level.
Widely known physical models are

i. Unifying model.
ii. Frame memory model.

2. **Object Based logical model** :

This model is used in describing data at logical and view level. Widely known models are : The Entity Relationship model, The Object Oriented model, The Semantic data model, The Functional data model.

3. **Record based logical model :**Record based logical models are used in describing data at the logical and view level.

   It is used to specify

   i. The overall logical structure of database.
   ii. To provide higher level description of implementation.

Record based logical models are so named because the data base is structured fixed format record. Each record type defines fixed number of fields or attributes and each field have usually of fixed length.

The three most widely accepted record based data models are

A) Relational Model
B) Network Model
C) Hierarchical Modal

A) Relational Model

This model uses a collection of tables to represent both data and the relationship among those data. Each table has multiple columns and each column has unique name.

Relational model is nothing but two dimensionaltables which contained Row's and columns.

e.g. Student Admission Record

Rollno	Name of Student	Class	Date of Admission	Feespaid	Subject
101	Rajan N. Chaudhari	TYBSc	10 July 2020	6700	Computer
102	Shaila v. Patil	TYBSc	12 June 2020	6700	Computer
234	Ram Nagar	Msc	10 August 2020	12500	Physics
456	Atul R. Patwardhan	MA	12 July 2020	4300	Politics

Advantages:

- Structural independence
- Conceptual simplicity.
- Design, implementation and maintenance easy.
- Use of query capability.

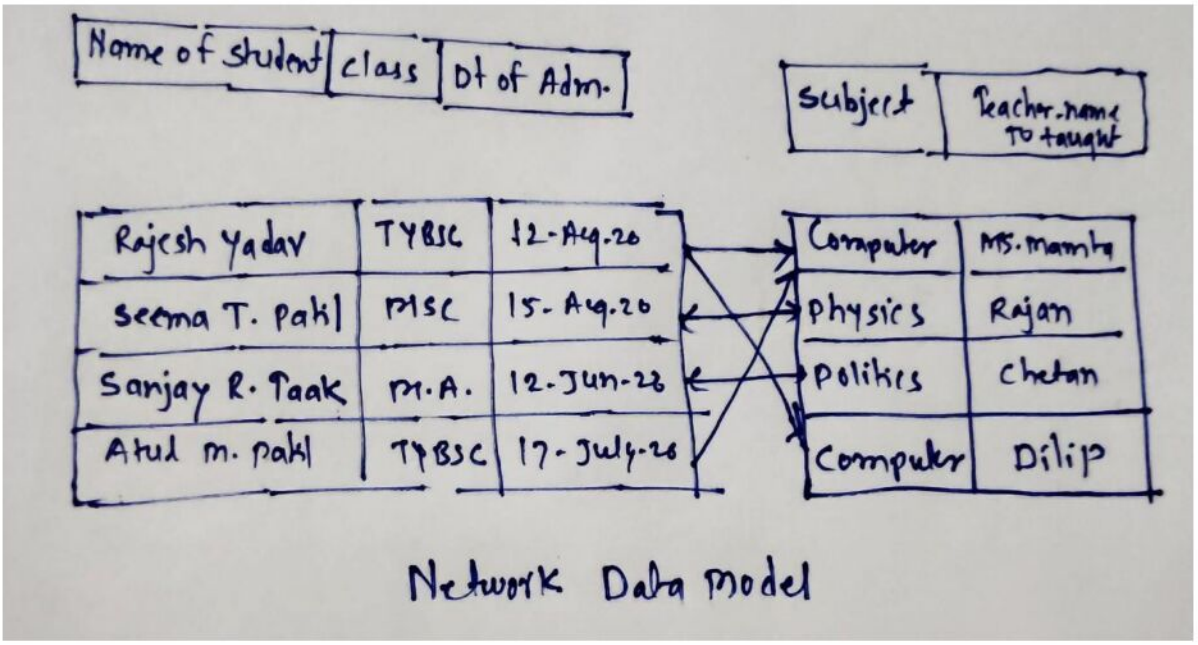
Disadvantages:

- Ease of design can lead to bad design.
- Hardware overheads.

B) Network Model :

Data in the network model are represented by collection of records and relationships among data are represented by links, also called as pointers.

The records in the database are organized as collection of arbitrary graph. This is shown in the following fig.



Advantages :

The main advantages of network model are :

- Conceptual simplicity.
- Capability to handle more relationship types.
- Easy for data access.
- Data integrity.
- Data independent.
- Database standards.

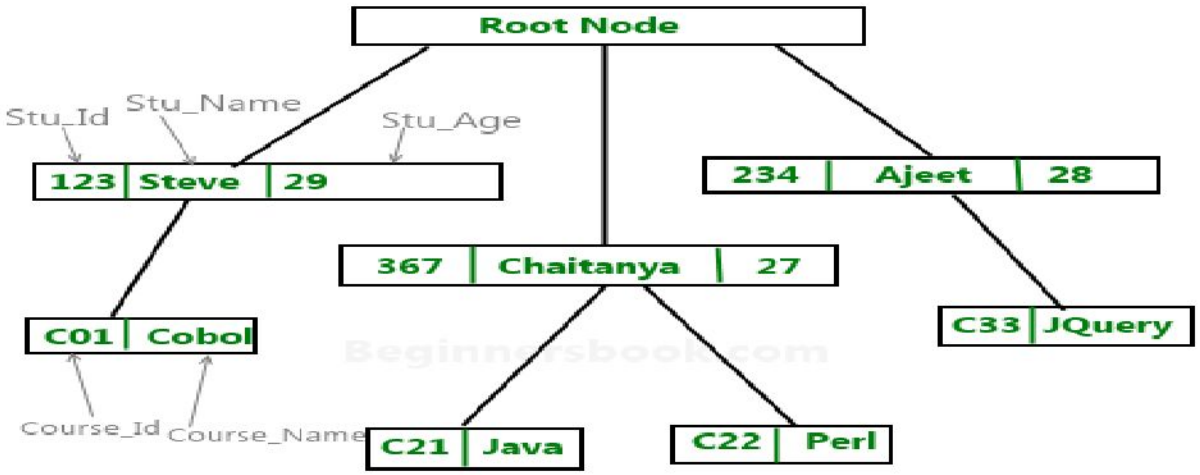
Disadvantages:

- System Complexity.
- Absence of structural independence.

C) **Hierarchical Model** :

The hierarchical model is similar to network model i.e. in which data and relationship among data are represented by records and links respectively. The records in the database are organized as collection of trees. This model does not handle many to much relationship.

It is shown in following fig.



Advantages

The hierarchical model had many advantages over the file system.

- Simplicity.
- Data security.
- Data integrity.
- Efficiency.

Disadvantages:

- Implementation Complexity.
- Database management problems.
- Lack of structural independence.
- Programming complexity.
- Implementation Limitation.

D) **Object Oriented Model :**

A data model consist of mathematical concept –

1. Static properties such as objects, attributes and relationships.
2. Integrity rules over objects and operations.
3. Dynamic properties such as operations or rules defining new database state.

Only Object oriented model have ability to integrate all these aspects i.e three components directly within the database and apply them uniformly to all users of the database. Which support for complex problems.
Object oriented model represents an entity as a class.

A class represents both object attributes as well as behavior of the entity.Object oriented databases manages objects. It is suited for multimedia applications as well as data with complex relationships.

Advantages :

- Handle large number of different data types.
- Use of Object oriented programming in database technology.
- Increases the productivity.
- Easy data access.

Disadvantages :

- Not suited for all applications.
- It is difficult to maintain.

Comparison of different Database models :

Model	Data Element Organization	Identity	Access Language
Hierarchical	Files, records	Record based	Procedural
Network	Files, Records	Record based	Procedural
Relational	Tables	Value based	Non - Procedural
Object Oriented	Objects	Record based	Procedural
Object Relational	Object infrastructure	Value based	Non-Procedural
Deductive	Fact, rule	Value based	Non-Procedural

A DBMS is a collection of interrelated files & set of programs which allows user to insert, modify, delete & access data from these files.'

A main purpose of DBMS system to provide abstract view of data means - system hides certain details of how data are stored & maintained.

1.4 Levels of data Abstraction :

Main purpose of DBMS system is to retrieve data efficiently. For that purpose design the structure of table or data base is very complex task. Because user of the database system not computer trained so developer of database system hide the complexity of database through several level of abstraction.

- **Physical level**

This is the lowest level of abstraction describes how data actually stored. At this level complex low-level data structure are described.

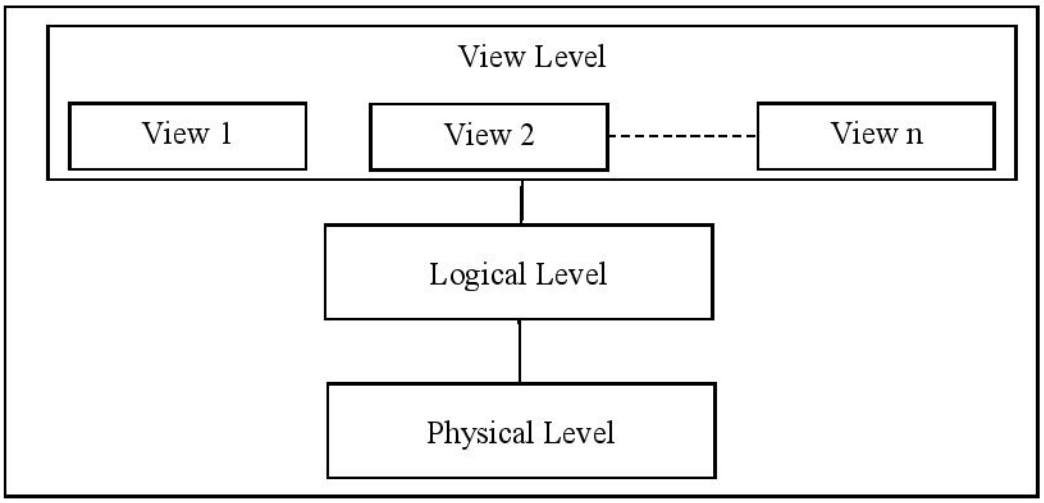
- **Logical level**

This is the next higher level of abstraction which describes what data are stored in the database, and what is the relationship exist among those data. The logical

level of abstraction is used by database administrators, who decide what information to be kept in the database.

- **View level**

The highest level of abstraction which describes only a part of the entire database. That is retrieval of information many way's. The system may provide many views for the same database.



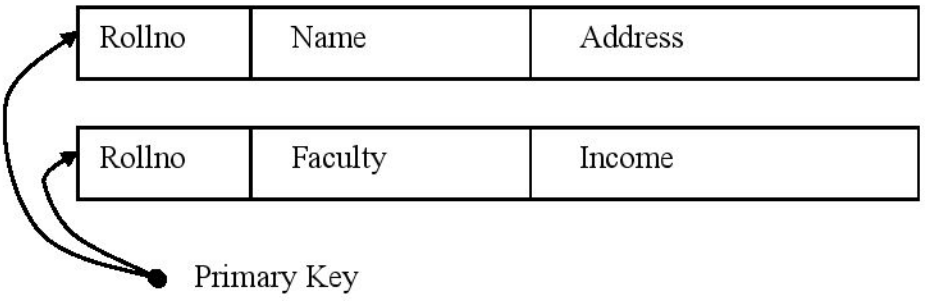
The three level of data abstraction is show in following figure.

Consider following example
Suppose we want t creates admission system of college.

- Physical level:-

Rollnoint
Name char
Faculty char
Income float
Address char

- Logical level:-



View level:-

1. List all the student of college.
2. List the entire student whose first name is Sam.
3. List all the male student of college.

Goal of 3 level/schema of Database

Here, are some Objectives of using Three schema Architecture:

- Every user should be able to access the same data but able to see a customized view of the data.
- The user need not to deal directly with physical database storage detail.
- The DBA should be able to change the database storage structure without disturbing the user's views
- The internal structure of the database should remain unaffected when changes made to the physical aspects of storage.

Instances :

Database change over time as information is inserted and deleted. The collection of information stored in the database at particular moment is called an instance of the database.

Value of variable in programming language is called instance. Eg.Int a=5;

Schemas :

The overall design of the database is called the *database schemas,* which changes frequently.

* Data independence:

The ability to modify a schema definition in one level without affecting a schema definition in the next higher level

There are two levels of data independence :

1.**Physical data independence**:Is the ability to modify the physical schema without causing application program to be rewritten.

2.**Logical data independence**:Is the ability to modify the logical schema without causing application program to be rewritten.

Queries in DBMS :

Query : A query is a statement requesting the retrieval of information. The portion of a DML which involves information retrieval is called query language.

SQL : SQL language is referred as a query language. It is also called asstandard relational database language. It is originally called as sequel. It is developed in early 1970s.

Database Language

A database system provides two different types of language.

i. To specify database schema.
ii. To express database queries.

Following are the categories of SQL DDL, DML, DCL and TCL

Data definition language (DDL)

A database schema is specified by a set of definition expressed byspecial language called data definition language (DDL).

Some of the SQL/DDL commands which are used in software

CREATE – to create database and its objects like (table, index, views, store procedure, function and triggers)

ALTER – alters the structure of the existing database

DROP – delete objects from the database

Data Manipulation Language (DML)

Data manipulation means

- Retrieval of information stored in the database.
- The insertion of new information into the database.
- The deletion of information from a database.
- The modification of information stored in the database.

The data manipulation language is a (DML) language that enables us to access, manipulate data.

SQL provides three data manipulation statements –and includes most common

SQL statements such **SELECT, INSERT, UPDATE, DELETE** etc,

Data Control Language (DCL) :

The data control language consist of commands that controls the user access to the database objects. Which is mainly related with security issues that is, determining who has access to the database object and what operations they can perform. The job of DCL to prevent unauthorized user to access data. The DCL commands are GRANT and REVOKE.

Transaction Control statement (TCL) :

Which manage all the changes made by the DML statements. Some of the transaction control statements are COMMIT,ROLLBACK,SAVEPOINT and SET TRANSACTION.

Users of DBMS :

There are different types of users which play different role in database environment.

1. **Application programmers** :
   Application programmer also called as Application developer. He is the person who is responsible for implement database applications. His responsibilities are :
   a. Design and develop the database application.
   b. Design the database structure for an application.
   c. Estimate storage.
   d. Specify modification.

2. **End user** : It is also called as database users. End user always interact with the database through application or utilities. End users responsibilities are :
   a. Perform data entry.
   b. Querying the database for decision making and other activities.
   c. Perform modifications and deletions of existing data.
   d. Generating reports, charts from the given data.

3. **Database Administrator** : Each database requires at least one database administrator (DBA) to administrate the database. DBA's responsibilities are :
   a. Installing and upgrading the DBMS and application tools.
   b. Backup and restore the database.
   c. Create primary database storage for database system.
   d. Modify database structure.
   e. Monitor user's access.
   f. Contacting DBMS vendors (seller's) for technical support.

4. **Sophisticated Users** interact with the system without writing programs :They form requests by writing queries in a database query language. These are submitted to a **query processor** that breaks a DML statement down into instructions for the database manager module.

5. **Specialized Users** are sophisticated users writing special database application programs. These may be CADD systems, knowledge-based and expert systems, complex data systems (audio/video), etc.

6. **Naive Users** are unsophisticated users who interact with the system by using permanent application programs (e.g. automated teller machine).

Advantages of DBMS

The database management system has a number of advantages as compared to traditional computer file-based processing approach. The DBA must keep in mind these benefits or

capabilities during databases and monitoring the DBMS.
The Main advantages of DBMS are described below.

- **Controlling Data Redundancy**

In non-database systems each application program has its own private files. In this case, the duplicated copies of the same data is created in many places. In DBMS, all data of an organization is integrated into a single database file. The data is recorded in only one place in the database and it is not duplicated.

- **Sharing of Data**

In DBMS, data can be shared by authorized users of the organization. The database administrator manages the data and gives rights to users to access the data. Many users can be authorized to access the same piece of information simultaneously. The remote users can also share same data. Similarly, the data of same database can be shared between different application programs.

- **Data Consistency**

By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users. If the DBMS has controlled redundancy, the database system enforces consistency.

- **Integration of Data**

In Database management system, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). This makes easy to retrieve and update data.

- **Integration Constraints**

Integrity constraints or consistency rules can be applied to database so that the correct data can be entered into database. The constraints may be applied to data item within a single record or the may be applied to relationships between records.

- **Data Security**

Form is very important object of DBMS. You can create forms very easily and quickly in DBMS. Once a form is created, it can be used many times and it can be modified very easily. The created forms are also saved along with database and behave like a software component. A form provides very easy way (user-friendly) to enter data into database, edit data and display data from database. The non-technical users can also perform various operations on database through forms without going into technical details of a fatabase.

- **Report Writers**

Most of the DBMSs provide the report writer tools used to create reports. The users can create very easily and quickly. Once a report is created, it can be used may times and it can be modified very easily. The created reports are also saved along with database and behave like a software component.

- **Control Over Concurrency**

In a computer file-based system, if two users are allowed to access data simultaneously, it is possible that they will interfere with each other. For example, if both users attempt to perform update operation on the same record, then one may overwrite the values recorded by the other. Most database management systems have sub-systems to control the concurrency so that transactions are always recorded with accuracy.

- **Backup and Recovery Procedures**

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damage due to failures to the computer system or application program. It is very time consuming method, if amount of data is large. Most of the DBMSs provide the 'backup and recovery' sub-systems that automatically create the backup of data and restore data if required.

- **Data Independence**

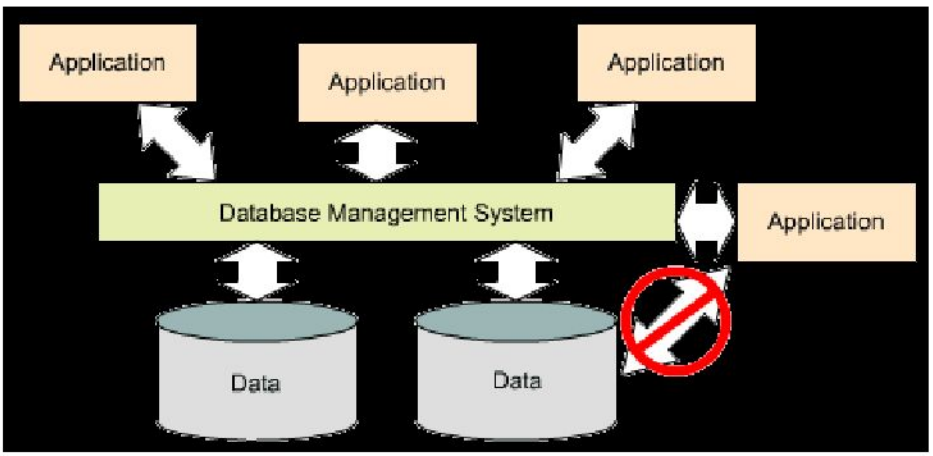
The separation of data structure of database from the application program that uses the data is called data independence. In DBMS, you can easily change the structure of database without modifying the application program.

Disadvantages of DBMS :

1. Cost of Hardware and Software : A processor with high speed of data processing and memory of large size is required to run the DBMS software. Similarly DBMS software is also very costly.

2. Cost of data conversion :
   When we want to convert traditional file system to database format, we require to convert data from file system to database.

3. Cost of staff training :
   Training required at all levels that is programming, application development and database administrator.
4. Technical staff :
   The trained technical person like database administrator (DBA), application programmer etc. are required to handle DBMS.
5. Database damage :
   In most of the organizations all the data is integrated into a single database. If database is damage due to any failure like electric, hard disk failure etc. due to this database may be corrupted.



This set of Database Questions & Answers focuses on “Relational Database and Database Schema”

1. A relational database consists of a collection of

a) Tables

b) Fields

c) Records

d) Keys

Answer:a
Explanation:Fields are the column of the relation or tables.Records are each row in relation.Keys are the constraints in a relation .

2. A _____ in a table represents a relationship among a set of values.

a) Column

b) Key

c) Row

d) Entry

Answer:c
Explanation:Column has only one set of values.Keys are constraints and row is one whole set of attributes.Entry is just a piece of data.

3. The term _____ is used to refer to a row.

a) Attribute

b) Tuple

c) Field

d) Instance

Answer:b
Explanation:Tuple is one entry of the relation with several attributes which are fields.

4. The term attribute refers to a _____ of a table.

a) Record

b) Column

c) Tuple

d) Key

Answer:b
Explanation:Attribute is a specific domain in the relation which has entries of all tuples.

5. For each attribute of a relation, there is a set of permitted values, called the _____ of that attribute.

a) Domain

b) Relation

c) Set

d) Schema

Answer:a
Explanation:The values of the attribute should be present in the domain.Domain is a set of values permitted .

Questions:

1. What are elements of database?
2. What do you mean by data?
3. What is relationship in a database?
4. What is database schema?
5. What is database?
6. Why do we need database?
7. What are characteristics of data in database?
8. What is database management system (DBMS)?
9. What are the functions of DBMS?
10. Explain hierarchical data model
11. What are advantages and disadvantages of hierarchical model.?
12. Explain relational data model
13. What are advantages and disadvantages of relational model.?
14. Explain network data model
15. What are advantages and disadvantages of network model.
16. Explain network data model
17. What are advantages and disadvantages of network model.
18. Define : 1.Instance 2. Schema 3. Meta data.
19. What is data definition language.
20. What is data manipulation language.
21. Define 1.Data dictionary 2. Metadata 3. Query.
22. Explain data independence.
23. Enlist the names of DBMS software's
24. What are the advantages of DBMS?
25. What are the disadvantages of DBMS?
26. List and explain the users of DBMS
27. What is relational database?
28. What is hypermedia database?
29. What is Operational database?
30. What is distributed database?
31. What is end user database?
32. What is external database?
33. What is navigational database?

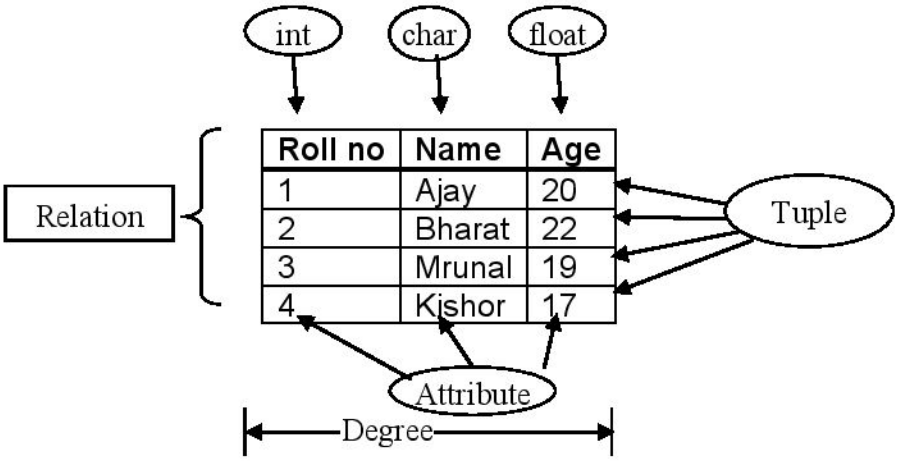
CHAPTER 2 :CONCEPTUAL DESIGN (E-R MODEL)

2.1 Overview of DB Design

The relational model is an abstract theory of data which is based on mathematical theory, introduced by Dr. E. F. Codd. The following table gives a list of relational terms & their corresponding informal equivalent.

Sr. No.	Formal relational terms	Informal equivalent
1	Relation	Table
2	Tuple	Row or record
3	Cardinality	Number of rows or records
4	Attribute	Column or field
5	Degree	No. of columns
6	Primary key	Unique identifier
7	Domain	Set of legal values or possible data values.

Consider following relational data structure.



The process of designing the general structure of the database:

Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.

Business decision – What attributes should we record in the database?

Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?

Physical Design – Deciding on the physical layout of the database

2.2 E-R Data Model :

The entity relationship model developed by Chen in 1976 for database designing.Which describe the structure of database & its associated retrieval &updates.

The basic concept of E-R model includes entity type, relationship type, & attributes
Entity type represents a set of objects in the real world with the same properties. .

Component of an E-R model :

E-R model forms basis for E-R Diagram, which represents conceptual view of database.

Entity :

The fundamental item in any data model is the entity. A database normally contains many different entity types, each has distinct set of attributes,and each entity has its own values or each attribute. consider example of book,information of student , authors details are example of an entity .

Entity Set :

An entity is a thing or object in the real world i.e. distinguishable from all other objects.
e.g. Each person is an enterprise in an entity.
An entity has a set of properties and the values for some set of properties may uniquely identity an entity.
e.g.
A student may have roll no. property that uniquely indentifies that student.
An entity set is a entities of the same type that sure the same properties, or attributes.
e.g. All the individual class students are the extension of the entity set student. Student contains the same category of attribute.

Roll No.	Name	Class
1	Ram	M. A.
2	Prerna	M. Com.
3	Anurag	M. Com.
4	Rajan	M. A.

And all the above attribute shares the common properties of student.

Attributes

The component of an entity is called attributes or a data item of an entity is called attribute.
An attribute is a single atomic unit of information which describes something about the named entity.

Attribute can have the same name in different entities
Each attribute is associated with set of values called as domain. The domain defines the potential value that an attribute many hold.

e.g. If the average age of the student of S.Y.B.Sc. Class is in between 15 to 20. Then we can define a set of values for the age attribute in the student entity.

Attribute may share a common domain. e.g. the date of birth attribute for student and teacher may be same, i.e. both attributes share common domain.

We can classify the attributes as following.

- **Simple**
- **Composite**
- **Single valued**
- **Multi valued**
- **Derived**

- Simple attribute

It is also called as atomic attribute. Simple attributes can not be further subdivided.
Consider e.g. of simple attribute for STUDENT entity,
rollno,sex, age are simple attribute.

- Composite attribute

An attribute composed of multiple components, each with an independent existence is called composite attribute.Some attribute can be further divided to yield or to get smaller components.

Consider entity STUDENT in which name and date of birth are composite attribute because name may be subdivided into firstname, lastmane,middlenamecomponents. Date of birth may be composed of into date-month-year [dd-mm-yy]. All these composed components are independent on each other.

- Single valued attribute

A single valued attribute is one that holds a single value for single entity.e.g. The classroom entity has single value for room number attribute, so room_no. attribute referred as single valued.

- Multi-valued Attribute

A multi-valued attribute is one which holds multiple values for a single entity.
e.g. Consider entity STUDENT can have multiple value for hobbyattribute : reading, music, movies, playing and so on.

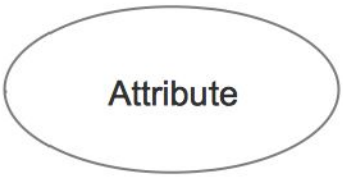
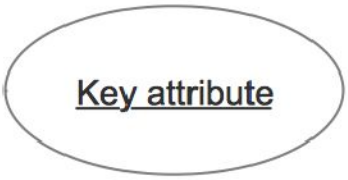
- Derived attribute

A derived attribute is one that represents a value which is derivable from the value of related attributes or set of attributes, not necessarily in the same entity.

Consider following e.g.

1. The age attribute can be derived from date of birth. Therefore they are related.
2. Consider PAY as entity in which Basic is a simple attribute and DA, HRA, TA, total are derived attributes.

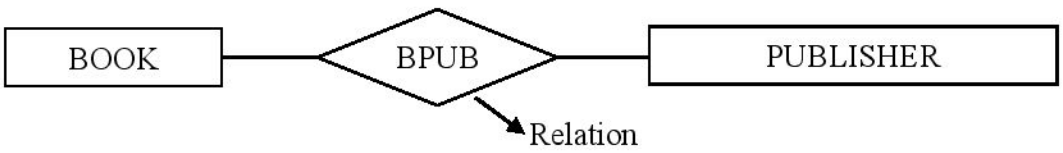
E-R diagram conventions

Symbol	Shape Name	Symbol Description
Entities		
	Entity	An entity is represented by a rectangle which contains the entity's name.
	Weak Entity	An entity that cannot be uniquely identified by its attributes alone. The existence of a weak entity is dependent upon another entity called the owner entity. The weak entity's identifier is a combination of the identifier of the owner entity and the partial key of the weak entity.
	Associative Entity	An entity used in a many-to-many relationship (represents an extra table). All relationships for the associative entity should be many
Attributes		
	Attribute	In the Chen notation, each attribute is represented by an oval containing atributte's name
	Key attribute	An attribute that uniquely identifies a particular entity. The name of a key attribute is underscored.
	Multivalued attribute	An attribute that can have many values (there are many distinct values entered for it in the same column of the table). Multivalued attribute is depicted by a dual oval.

	Derived attribute	An attribute whose value is calculated (derived) from other attributes. The derived attribute may or may not be physically stored in the database. In the Chen notation, this attribute is represented by dashed oval.
Relationships		
	Strong relationship	A relationship where entity is existence-independent of other entities, and PK of Child doesn't contain PK component of Parent Entity. A strong relationship is represented by a single rhombus
	Weak (identifying) relationship	A relationship where Child entity is existence-dependent on parent, and PK of Child Entity contains PK component of Parent Entity. This relationship is represented by a double rhombus.

The association or relationship which exit between the entities which relates data items in meaningful way. Relationships are represented by diamond shaped symbol with the relation name inside the diamond. The two sides of entities to which we want to relates.

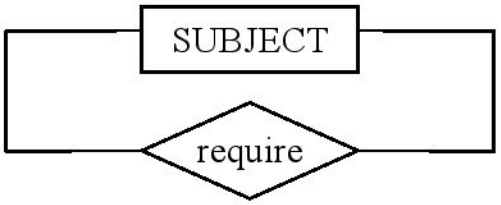
Consider following example : Suppose we have two entities Book and PUBLISHER are relates with each other, it is represented in the following way.



2.3 Conceptual design using ER

Degree

The degree of relationship indicates the no. of associated entities. A unary relationship exists when association is maintained within a single entity. Consider situation where to take a particular subject, which is shown in following fig.



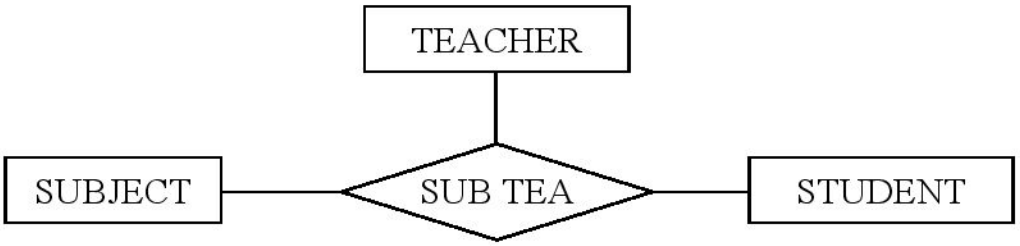
Unary Relationship

A binary relationship exists when two entities are associated. Consider the e.g. of BOOK & PUBLISHER entity.



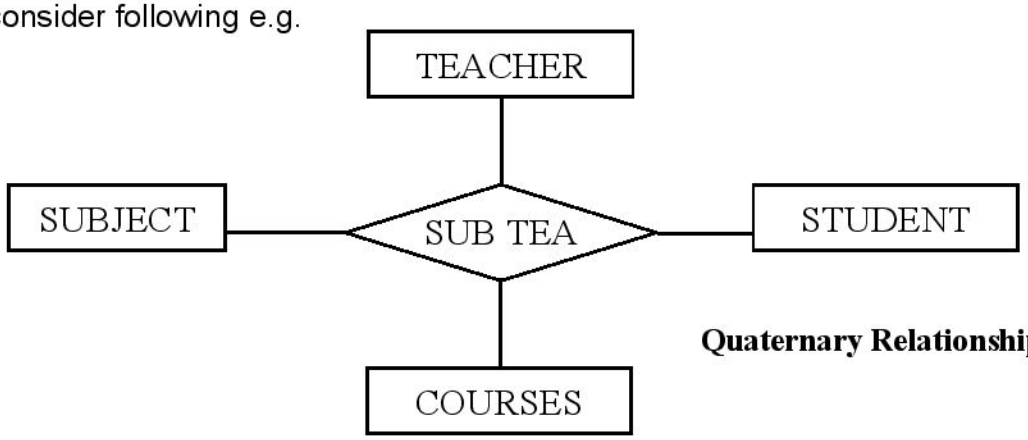
Binary Relationship

A Ternary relationship exist when three entities are associated consider the e.g.



Ternary Relationship

A quaternary relationship exists when there are four entities associated consider following e.g.



Quaternary Relationship

Connectivity :

Relationship can be classified as one to one, one to many & many to many which is shown or describe in following fig.

1)



One - to - One Relationship / Mapping

2)



One - to - Many Relationship / Mapping

3)



Many - to - Many Relationship / Mapping

Relationship Set:

A relationship is an association among several entities. A relationship set is a set of relationship of the same type

The relation always exists between more than one entity. A relationship may also have attribute called descriptive attributes. There can be more than one relationship set involving the same entity sets.

e.g. The relationship between customer and account describes as various relationships.

i) Customer can open account.

ii) Customer may withdraw amount from accont.

iii) Customer may deposit amount into accont and so on.

KEYS

We must have a way to specify how entities within a given entity set are distinguished. The values of the attribute values of an entity must be such that they can uniquely identify the entity. In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes.

A key allows us to indentify a set of attributes that suffix to distinguish entities from each other. Key also help uniquely identify relationship.

Primary Key:

A key in the entity set which having it's unique identification with the property of uniqueness.

The primary key values are rarely changeable. The function of the primary key is not only for the uniqueness purpose but also it can be used for relating the multiple entities with each other.

e.g. A student entity set contains rollno, name, address, class, date of admission.
Student (**rollno**, name, address, class, date of admission)
Here rollno is the primary key because it fulfils all the characteristics of primary key.

Foreign/Reference/Secondary Key:

In one of the very important attribute in the entity set. The existence of this attribute is located from another entity set and hence foreign key. The primary key is used in another table or entity set becomes a reference key. The importance or priority of this attribute is 2 and hence secondary key.
e.g.

P F/R/S
|
{employee (**emp_no**, name, job, **dept_no**)
dept (**dept_no**, name, location)

Super Key :

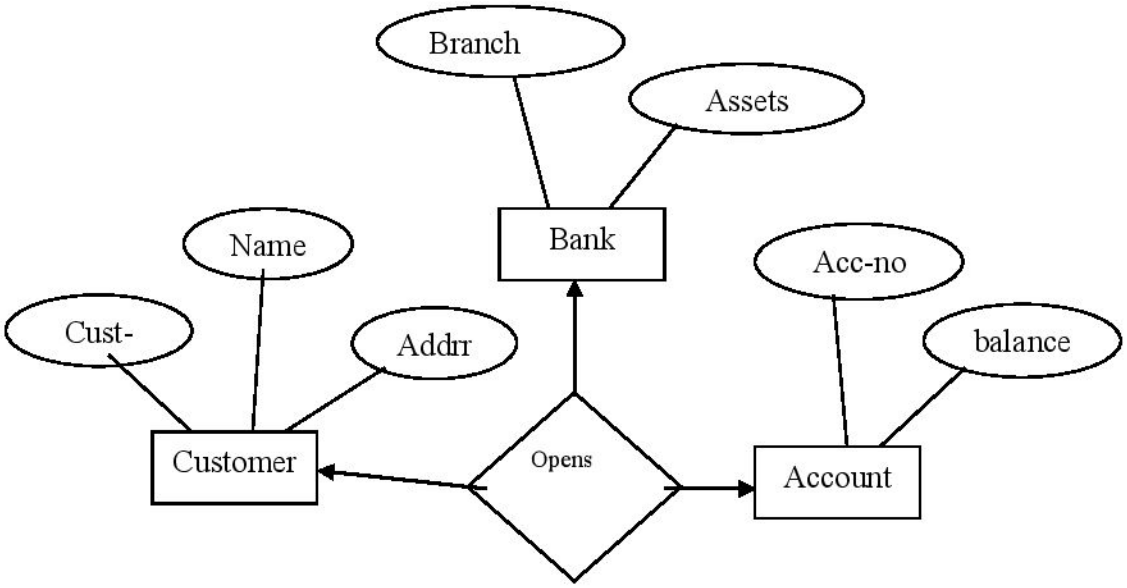
A super key is a set of one or more attributes that taken collectively allow us to identity uniquely in the entity set.
e.g. Consider, the student entity. In the student entity, the rollno is not the sufficient attribute to recognize or indentify the student, therefore it is necessary to provide some extra attribute like (Rollno, name, address)
To identify the student easily, the group of (rollno, name, address) is known as super key.

Candidate keys:

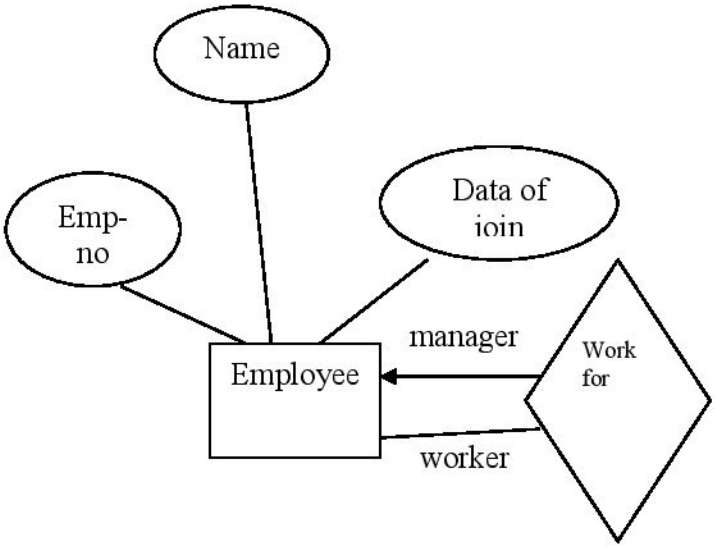
As per the above discussion in the super key when we prepare multiple set of attributes for recognizing student are the candidates.
e. g. : From student, the candidate key are as follows:-

i) Rollno, name, address
ii) Rollno, class
iii) Rollno, address, class

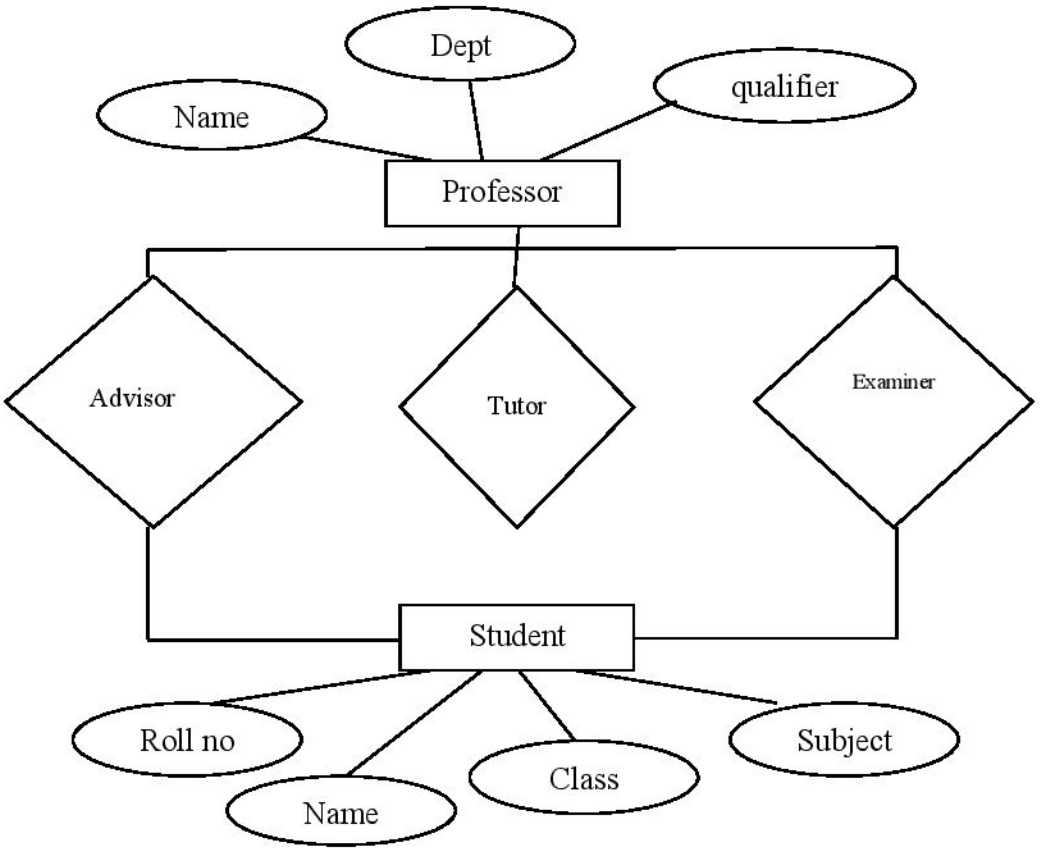
E-R Diagram more than two Entities



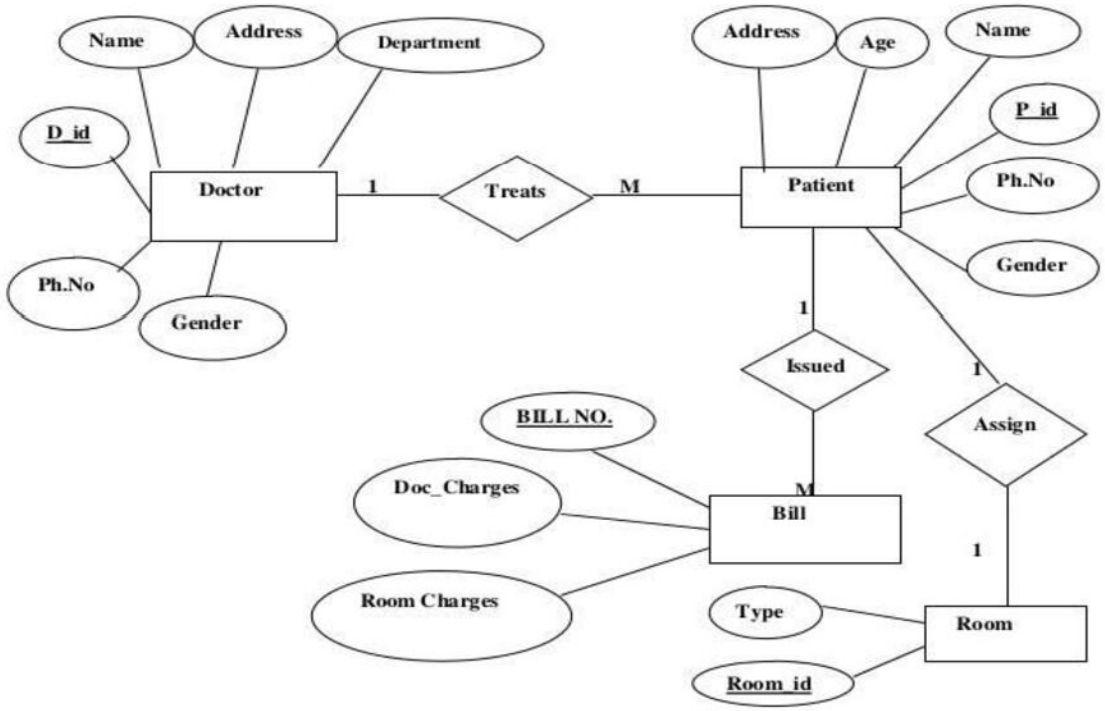
Unary ERD

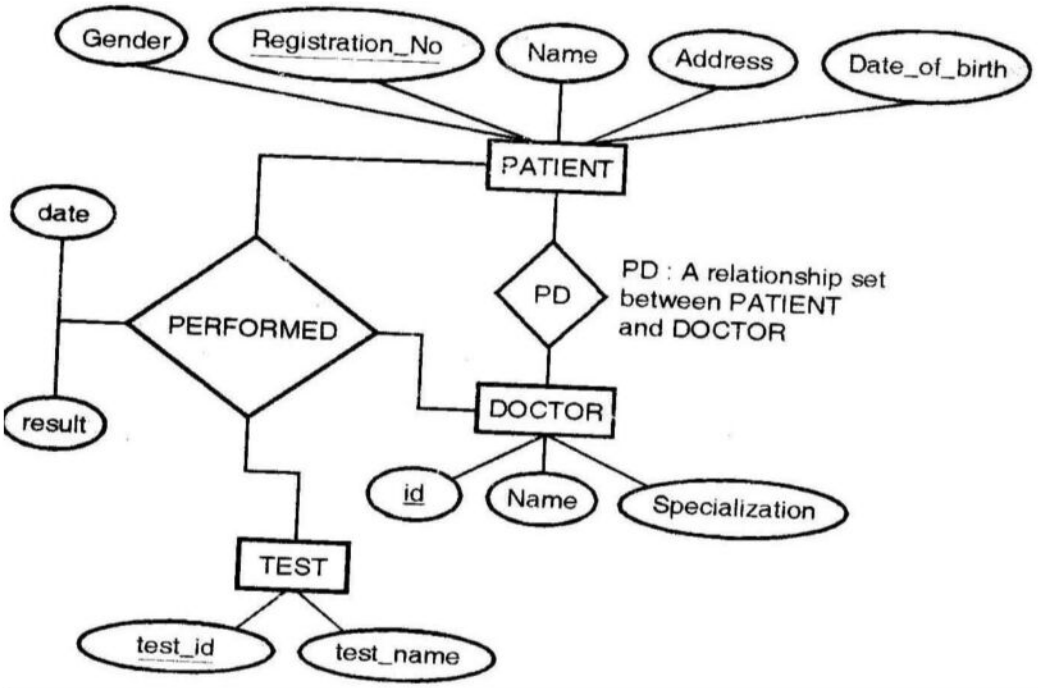


Role indicator ERD:



Hospital Management system





MCQ on "The Entity-Relationship Model"

1. An _____ is a set of entities of the same type that share the same properties, or attributes .

a) Entity set

b) Attribute set

c) Relation set

d) Entity model

Answer:a
Explanation:An entity is a "thing" or "object" in the real world that is distinguishable from all other objects.

2. Entity is a

a) Object of relation

b) Present working model

c) Thing in real world

d) Model of relation

Answer:c
Explanation:For example, each person in a university is an entity.

3. The descriptive property possessed by each entity set is _____ .

a) Entity

b) Attribute

c) Relation

d) Model

Answer:b
Explanation:Possible attributes of the instructor entity set are ID, name, dept name, and salary.

4. The function that an entity plays in a relationship is called that entity's _____.

a) Participation

b) Position

c) Role

d) Instance

Answer:c
Explanation:A relationship is an association among several entities.

5. The attribute *name* could be structured as a attribute consisting of first name, middle initial, and last name . This type of attribute is called

a) Simple attribute

b) Composite attribute

c) Multivalued attribute

d) Derived attribute

Answer:b
Explanation:Composite attributes can be divided into subparts (that is, other attributes).

6. The attribute AGE is calculated from DATE_OF_BIRTH . The attribute AGE is

a) Single valued

b) Multi valued

c) Composite

d) Derived

Answer:d
Explanation:The value for this type of attribute can be derived from the values of other related attributes or entities.

7. Not applicable condition can be represented in relation entry as

a) NA

b) 0

c) NULL

d) Blank Space

Answer:c
Explanation:NULL always represents that the value is not present.

8. Which of the following can be a multivalued attribute ?

a) Phone_number

b) Name

c) Date_of_birth

d) All of the mentioned

Answer:a
Explanation:Name and Date_of_birth cannot hold more than 1 value.

9. Which of the following is a single valued attribute

a) Register_number

b) Address

c) SUBJECT_TAKEN

d) Reference

Answer:a
Explanation:None.

10. In a relation between the entities the type and condition of the relation should be specified . That is called as_____attribute

a) Desciptive

b) Derived

c) Recursive

d) Relative

Answer:a
Explanation:Consider the entity sets student and section, which participate in a relationship set takes. We may wish to store a descriptive attribute grade with the relationship to record the grade that a student got in the class.

11. Which of the following gives a logical structure of the database graphically ?

a) Entity-relationship diagram

b) Entity diagram

c) Database diagram

d) Architectural representation

Answer:a
Explanation:E-R diagrams are simple and clear—qualities that may well account in large part for the widespread use of the E-R model.

12. The entity relationship set is represented in E-R diagram as

a) Double diamonds

b) Undivided rectangles

c) Dashed lines

d) Diamond

Answer:d
Explanation:Dashed lines link attributes of a relationship set to the relationship set.

13. The Rectangles divided into two parts represents

a) Entity set

b) Relationship set

c) Attributes of a relationship set

d) Primary key

Answer:a
Explanation:The first part of the rectangle , contains the name of the entity set. The second part contains the names of all the attributes of the entity set.

14. Consider a directed line(->) from the relationship set advisor to both entity sets instructor and student. This indicates _____ cardinality

a) One to many

b) One to one

c) Many to many

d) Many to one

Answer:b
Explanation:This indicates that an instructor may advise at most one student, and a student may have at most one advisor.

15. We indicate roles in E-R diagrams by labeling the lines that connect _____ to _____.

a) Diamond , diamond

b) Rectangle, diamond

c) Rectangle, rectangle

d) Diamond, rectangle

Answer:d
Explanation:Diamond represents a relationship set and rectangle represents a entity set.

16. An entity set that does not have sufficient attributes to form a primary key is termed a _____ .

a) Strong entity set

b) Variant set

c) Weak entity set

d) Variable set

Answer:c
Explanation:An entity set that has a primary key is termed a strong entity set.

17. For a weak entity set to be meaningful, it must be associated with another entity set, called the

a) Identifying set

b) Owner set

c) Neighbour set

d) Strong entity set

Answer:a
Explanation:Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be existence dependent on the identifying entity set. The identifying entity set is said to own the weak entity set that it identifies.It is also called as owner entity set.

18. Weak entity set is represented as

a) Underline

b) Double line

c) Double diamond

d) Double rectangle

Answer:c
Explanation:An entity set that has a primary key is termed a strong entity set.

19. If you were collecting and storing information about your music collection, an album would be considered a(n) _____.

a) Relation

b) Entity

c) Instance

d) Attribute

Answer:b
Explanation:An entity set is a logical container for instances of an entity type and instances of any type derived from that entity type.

20. What term is used to refer to a specific record in your music database; for instance; information stored about a specific album?

a) Relation

b) Instance

c) Table

c) Column

Answer:b
Explanation:The environment of database is said to be instance. A database instance or an 'instance' is made up of the background processes needed by the database.

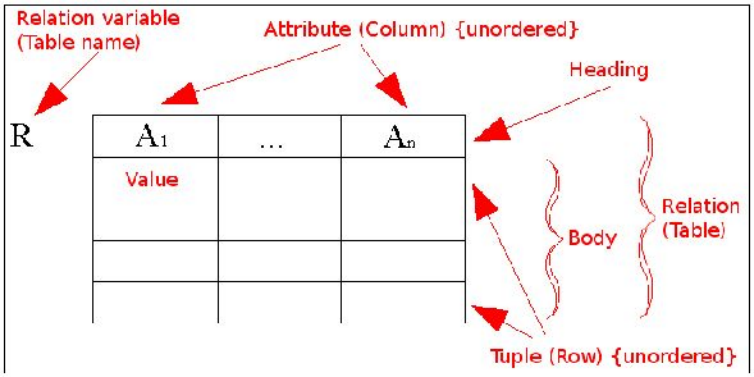
Questions :

1. What is E-R diagram.
2. What are components of an E-R diagram.?
3. What are entities.?
4. What are attributes?
5. What is domain?
6. What are different types of attributes?
7. What is simple attribute? Give example.
8. What is composite attribute? Give example.
9. What is single valued attribute? Give example.
10. What is multi-valued attribute? Give example.
11. What is derived attribute ? Give example.
12. What are conventions for representing entities & attributes in E-R diagram?
13. What are relationship?
14. What do you mean by degree of relationship?
15. What are unary relationship?
16. What are binary relationship?
17. What are ternary relationship?
18. What are quaternary relationship?
19. What do you mean by cardinality of a relationship.?
20. What are strong and weak entities?
21. What do you mean by connectivity of a relationship?
22. Differentiate weak and strong entity set.
23. Construct E-R diagram for a car insurance company that has set of customers, each one has one or more cars. Each car has associated with it zero or any number of recorded accidents.
24. Construct E-R diagram for a hospital with set of patients and set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.
25. Draw E-R diagram for Examination system of University.
26. Draw E-R diagram of Admission system of College.
27. What is entity set? Explain different types of attributes while constructing entities.

3. Relational Data Model

3.1 Relations (Concepts, Definition)

A **relation** is a data structure which consists of a heading and an unordered set of tuples which share the same type.
Relation is a fundamental concept in relational model.
A relation has zero or more tuples.



Relation Variable – Different names of relations may combine or relates with each other

Attribute : One of the characteristic of relation

Value : Value of Attribute or column

Tuple : A complete row which contains all attribute values

Example :

Consider the relation Student contains

Rollno	Name	Class	Dtofadm
101	Rajesh Patel	FYBSC	10-JUN-2011
102	PritamPande	FYBSC	13-JUL-2011
103	NilamRane	FYBSC	16-AUG-2011

3.2Conversion of ER to Relational Model :

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram

Convert each strong entity to a relation.

Employee
Emp_no
ename
Address
Salary
DtofJoin
Dept_no

Department
Dept_no
Dname
Location

Employee (Emp_no, ename, Address,salary,Dtofjoin,Dept_no)
Department(Dept_no,Dname,Location)

- Attributes of the entity type become attributes of the relation.
- Include only simple attributes in relation. For composite attributes, only create attributes in the relation for their simple components.
- Multi-valued attributes are handled separately
- The primary key of the relation is the key attributes for the entity

Employee (**Emp_no**, ename, Address, salary, Dtofjoin, **Dept_no**)

Department(**Dept_no**,Dname,Location)

- Convert binary 1:1and 1:Nrelationships into a UNIQUE foreign key reference from one relation to the other.
- Given a binary 1:1 relationship *R* between two entities *Ei*and*Ej*:
    1. Identify the corresponding relations *Ri*and*Rj*.
    2. Chose one of the relations, say *Ri*, and:
        - Add the attributes of *R* to *Ri*.
        - Add the primary key attributes of *Rj*to*Ri*, and create a foreign key reference to *Rj*from*Ri*.
        - Declare these primary key attributes of *Rj*to be UNIQUE

- Given a binary 1:N relationship *R* between two entities *Ei*and*Ej*:
    - Identify the corresponding relations *Ri*and*Rj*.
    - Let *Ri*be the N-side of the relation.
    - Add the attributes of *R* to *Ri*.
    - Add the primary key attributes of *Rj*to*Ri*, and create a foreign

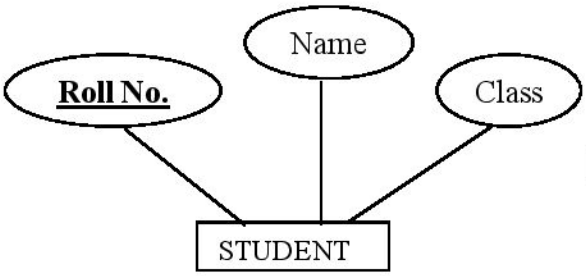
key reference to *Rj*from*Ri*.

By using above concept about converting ERD to Relational Model(Table) following tips should be remember.

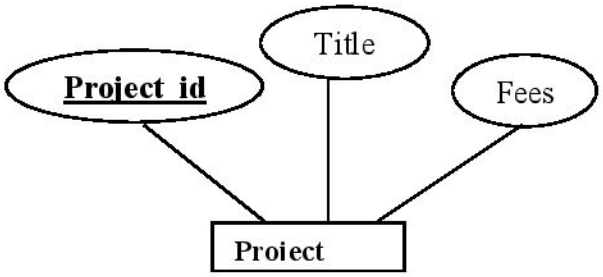
- An Entity set can be represented by a table with a column for every attribute of the entity set.
- If the Entity set under consideration is a weak entity set then there is a column in the table for every key attribute of the second entity set.
- A relationship set among entity sets E1,E2 .. is represented by a table whose columns consist of the keys for each of E1,E2…

Converting E-R Diagram into Tables

I] Converting Entity set into Tables



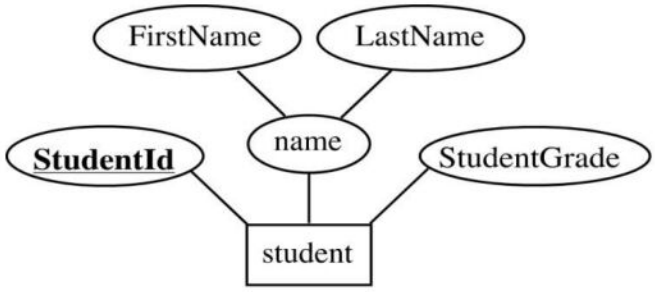
STUDENT		
Roll_no	Name	Class

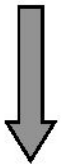


PROJECT		
Project_id	Title	Fees

II] Converting Entity with Composite attribute set into Tables

The Entity **Student** contains name attribute but the name attribute is composite attribute with two attributes FirstName and Last Name will represent as follows . While converting this diagram in tables the table looks like as follows.



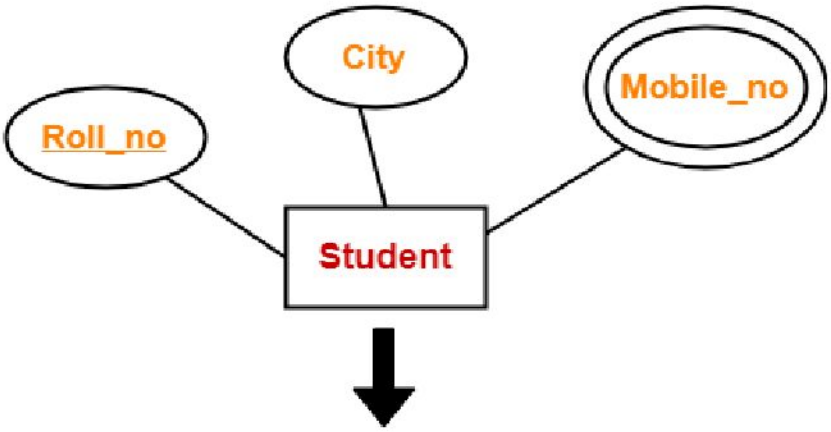


	STUDENT		
Roll_no	FirstName	LastName	StudentGrade

III] For Strong Entity Set With Multi Valued Attributes-

A strong entity set with any number of multi valued attributes will require two tables in relational model.

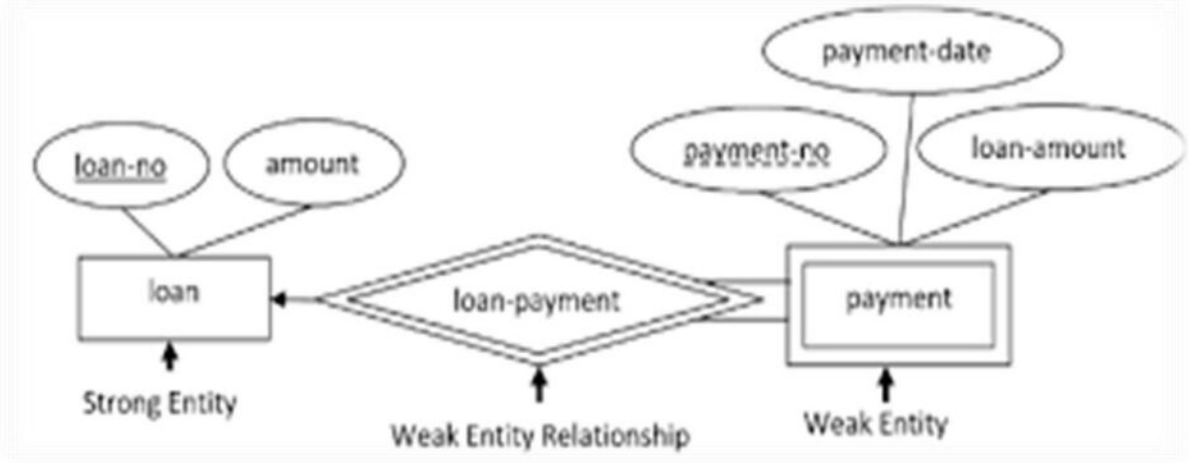
- One table will contain all the simple attributes with the primary key.
- Other table will contain the primary key and all the multi valued attributes.



Roll_no	City

Roll_no	Mbile_no

IV] Representing Weak Entity Sets as a Table

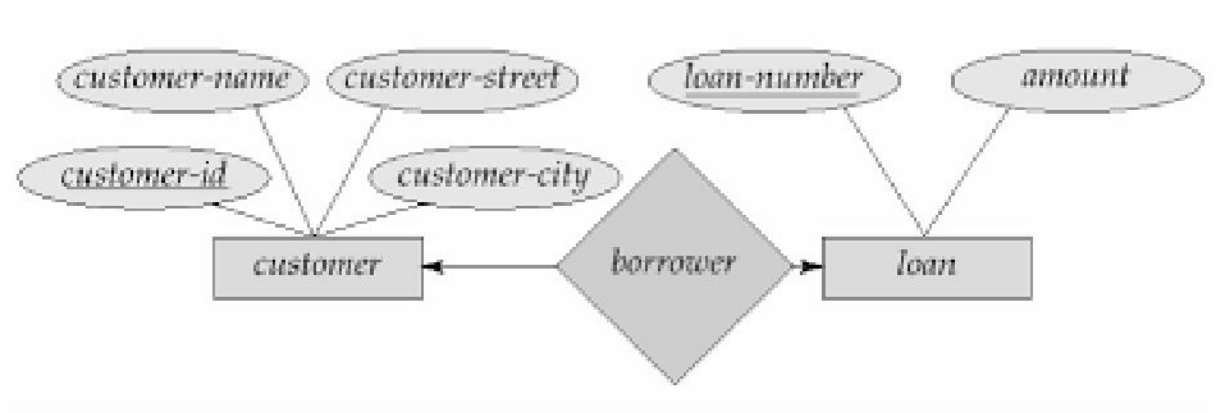


Include the Primary key of the linked strong entity set and all the attributes of Weak Entity Set, the result table looks like as follows.

Loan_no	Payment_no	Payment_date	Loan_amount

V] Representing relationship Sets as Tables

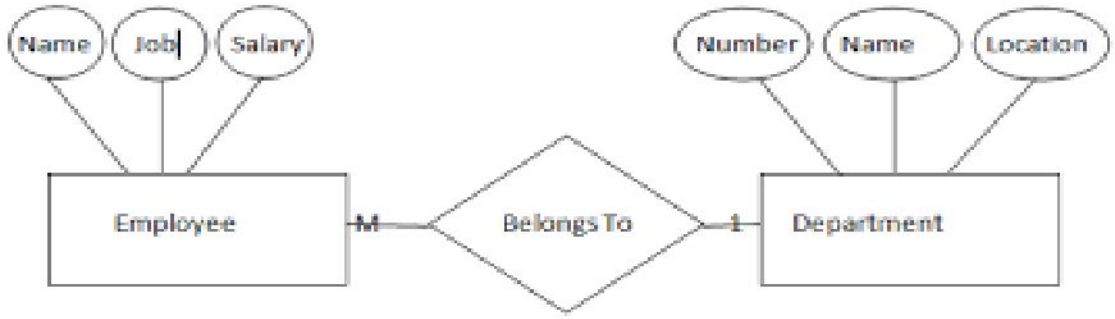
Many-to-Many Relationship



In the above diagram **Customer and Loan** both are entities and their relationship is **Many-to-Many**. While converting this diagram in to table looks as follows.

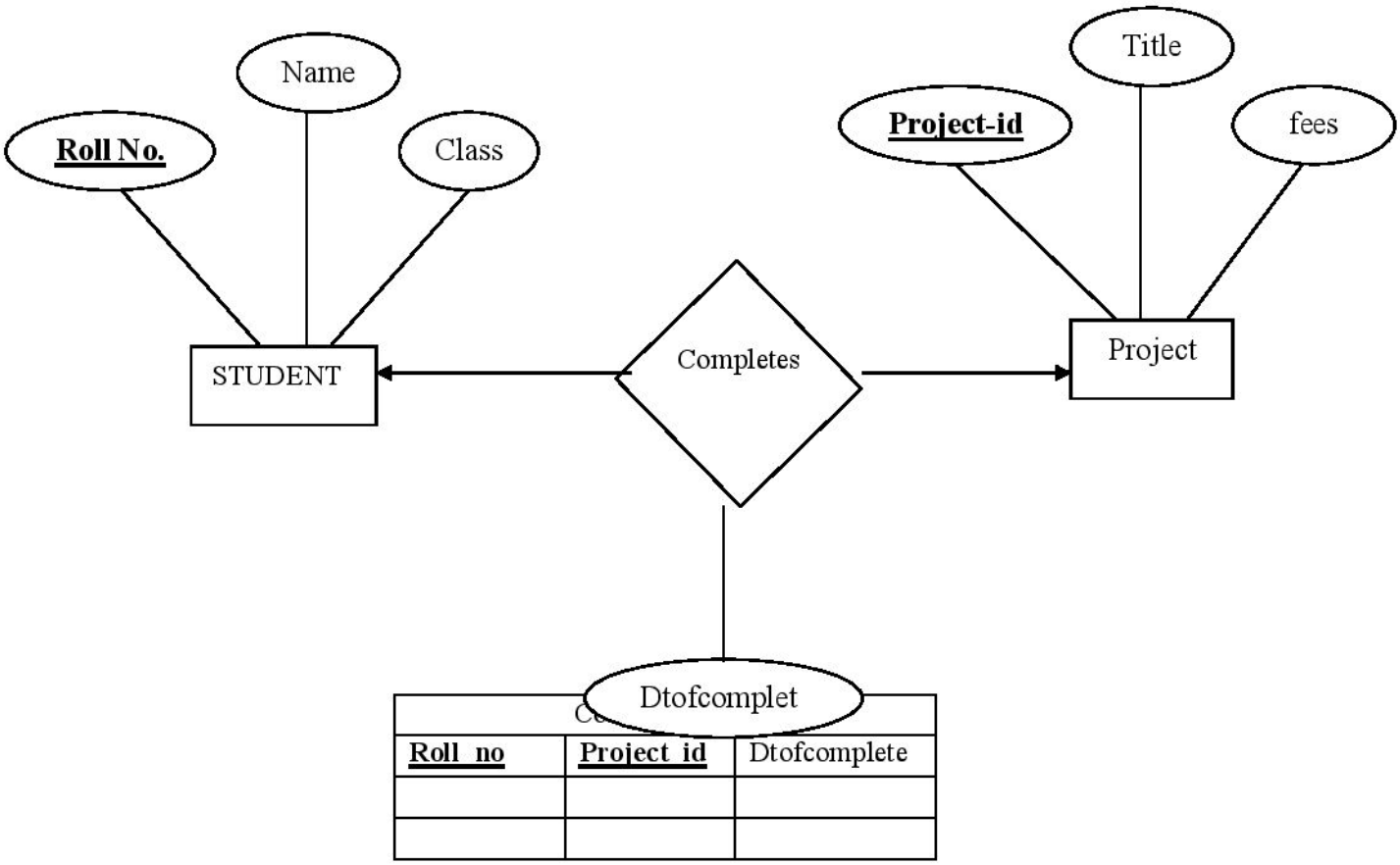
Customer-id	Loan_number

Many-To-One One-To-Many or Relationship

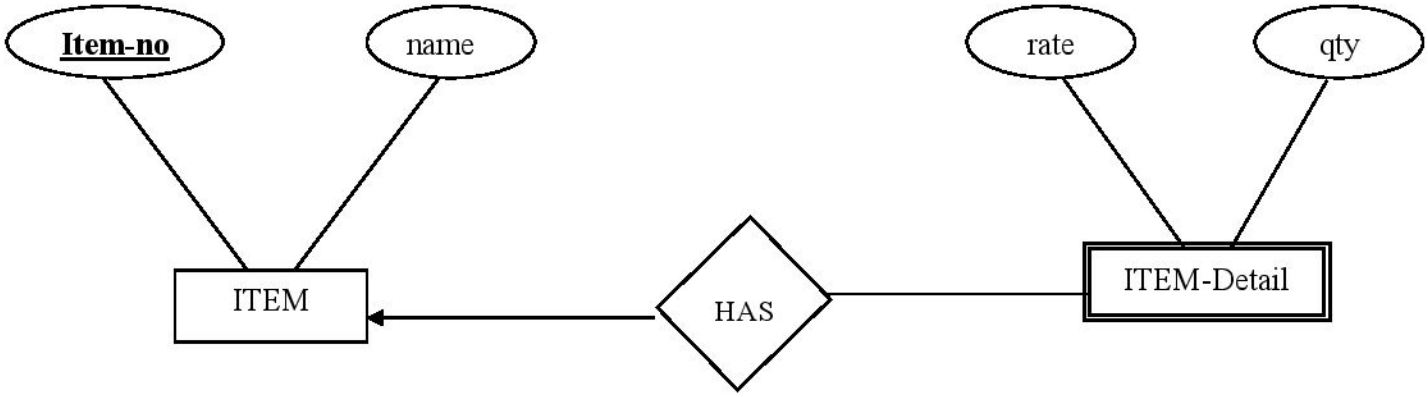


In the above E-R Diagram **Employee and Department** this relationship is many to one . While converting this diagram in to tables . Take all the columns of Employee and Primary key of Department as and for the One to Many relationship again apply the same method.

Name	Job	Salary	**Number**



Roll_no	Project_id	Dtofcomplete



ITEM_DETAILS			
Item_no	Name	**Rate**	Qty

3.3 Integrity constraints

Integrity constraints provide a means of changes made to the database by unauthorized user or authorized users do not result in a loss of data consistency. So integrity constraints prevent accidental damage of the database.
When the contents of database are modified with INSERT,DELETE OR UPDATE statement, the integrity of the data can be lost in many different ways.

- Invalid data may get added to the database.
- Changes to the database may lost due to power failure or crash.
- Existing data may be modified to an incorrect value.
- Parent rows could get deleted when child rows exist. e.g. Department code get deleted , when there are still employees belonging to the department in employee table.

Domain Constraints

Domain Constraints specify or restrict the set of possible values for a particular column. One example of this is prohibiting NULL as a value for a column in the table.

Use a domain in the column definition of **CREATE TABLE** or **ALTER TABLE ADD COLUMN** statements in place of a data type to define the set of valid values entered in the column.

CREATE DOMAIN*column-name datatype*
CONSTRAINT*constraint-name*
CHECK*predicate*

column-name - Name of the column you want to specify a domain for.
datatype - Datatype of the column *column-name*.
constraint-name - Name you want to give to this domain constraint.
predicate - Boolean expression used to define the domain

EXAMPLES

We have a payroll database that holds information on employees. One piece of such information is the employee's wage. We never want to allow values less than minimum wage in the wage field so we create the constraint:

CREATE DOMAIN*wage* **NUMERIC(5,2)**
CONSTRAINT*minimum-wage-test*
CHECK(VALUE>= 6.50)

Many times you do not want to allow NULL values in a column. To restrict NULL values just make the *predicate* of the **CHECK** clause = (**VALUE NOT NULL**).

It is also possible to explicitly declare the values that can be included:

CHECK(CITY IN("JALGAON", "DHULE"))

Examples :

CREATE DOMAINaccount_typechar(10)
CONSTRAINTacct_type_test (value in ("saving", "current", "loan"))

CREATE DOMAINclass_typechar(8)
CONSTRAINTclass_type check (value in ("F.Y.", "S.Y.", "T.Y."))

Referential Integrity -

Referential Integrity constraints tests for at least two relations. It primarily used for Foreign Keys .An attribute of a relation that is prime attribute of another relation is called a foreign key .

The concept of Referential Integrity constraint is defined as follows
Employee

Emp_no	Ename	City	Salary	Deptno	DtofJoin
2345	Ramprasad	Jalgaon	34567	10	12/10/1990
3456	Khilchand	Dhule	56780	20	21/03/1978
5673	Rohit	Jalgaon	45678	10	22/08/2000
2789	Saroj	Dhule	67890	30	20/05/2001

Department

Deptno	Dname	Location
10	Marketing	Rat Kila
20	Finance	22 pali
30	Computer	Church Area

Here the column Department(Deptno) is the Primary key and the same key is initiated in Employee(Deptno) as a Foreign Key or Reference Key. The Integrity holds due to relationship of references between two tables due to foreign key.
Referential Integrity protects against Insertion, Updation and Deletion operation between multiple relations as
Insertion : While inserting record in reference table it is mandatory to insert primary key valued record in prime table.
Update : There are two cases for updating records

i) In updating the record of table in which primary key is defined if related with other table with the reference of primary key can not directly update.

ii) While updating reference table first you have to update related value of primary key

Delete : Before deleting record from prime table it is necessary to delete all the referenced records of key from reference table.

Key Constraints :

1. Primary Key:

The primary key of a database table is a unique identifier assigned to each record. Each table will have one or more columns designated as the primary key.

Example:

Consider the table Student with Exam Data

Student(**Seatno**, Name, Class)

Here Seatno is uniqueidentifier and it can possible through only by the constraint **Primary Key**

Example:

Seatno Numeric(6) PRIMARY KEY

The simplest way to enforce referential integrity is not to allow changes to a primary key.

2. Foreign Key(Reference Key)

A foreign key is an identifier in a table that matches the primary key of a different table.

The foreign key creates the relationship with a different table, and referential integrity refers to the relationship between these tables.

When one table has a foreign key to another table, the concept of referential integrity states that you may not add a record to the table that contains the foreign key unless there is a corresponding record in the linked table.

It also includes the techniques known as cascading update and cascading delete, which ensure that changes made to the linked table are reflected in the primary table.

Example

An Employee can place one or more than order in market for purchasing different products as in the example follows

Employee(**EmpId**, Name, Address,City)
Order(**OrderId**, Date, ProductId,Qty,**EmpId**)

In the above tables EmpId,OrderId are Primary keys and EmpId in Order table is Foreign key which establishes the relationship between these tables.

3.4Codd's Rule :

E.F Codd was a Computer Scientist who invented Relational model for Database management. Based on relational model, Relation database was created. Codd proposed 13 rules popularly known as Codd's 12 rules to test DBMS's concept against his relational model. Codd's rule actually define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS).. The Codd's 12 rules are as follows.

Rule zero :

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

Rule 1 : Information rule

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

Rule 2 : Guaranteed Access

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

Rule 3 : Systematic treatment of NULL

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

Rule 4 : Active Online Catalog

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

Rule 5 : Powerful language

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

Rule 6 : View Updation rule

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

Rule 7 : Relational Level Operation

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

Rule 8 : Physical Data Independence

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

Rule 9 : Logical Data Independence

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

Rule 10 : Integrity Independence

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

Rule 11 : Distribution Independence

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

Rule 12 :Nonsubversion rule

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

Functional Dependency

Functional dependency describes the relationship between attributes. Functional dependency arises in many ways. If R represents a relation whose attributes are A1, A2,…An , and X is a set of attributes that uniquely determines the values of attributes in Y then we say,

X is functionally determines Y and is denoted by $X \rightarrow Y$, and Functional Dependency (FD) denoted by $X \rightarrow Y$

Consider following example

R= **STUDENT** : ROLLNO, NAME, ADDRESS, PLOT_NO

Value of ROLLNOdetermines, the value of student name. similarly address also functionally depends on ROLLNO. But address may be functionally dependant on a set of attribute. i.e. address is functionally dependent on student ROLL + NAME as well as PLOT_NO.

In terms of FD we can say

ROLLNO →NAME
ROLLNO →ADDRESS
ROLLNO →PLOT_NO

These FD combined together :

STUDENT→ ROLLNO, NAME, ADDRESS, PLOT_NO

Also the FD creates as

ADDRESS→PLOT_NO

Closure of a Set of Functional Dependencies:

It is not sufficient to consider the given set of function dependency directly; we can prove that certain other functional dependency using some rules of closure set. The set of functional dependencies that can be derived from a given set of functional dependencies F is called the closure F. **Closure of F is written as F^+**

Consider following example :

Suppose that the schema R = (A, B, C, G, H, I) and the set of function dependencies

A → B
A → C
CG → H
CG → I
B → H

The functional dependency, A→ H, is logically implied.

Must hold t_1 [A] = t_2 [A], we have given that A → B, from this function dependency we have t_1 [B] = t_2 [B].

Since we have also given that B → H so t_1 [H] = t_2 [H]. Hence t_1 and t_2 are tuples such that t1[A]=t2[A] therefore t1[H]=t2[H] holds. Such functional dependency is called logical implied functional dependency from functional dependency set F, called closure of F i.e. F^+.

Armstrong's axioms are as follows:

The computation of F^+ depends on various rules which are listed below.

- Reflexivity rule : If $\propto$ is a set of attributes and $\beta \subseteq \propto$ then $\propto \rightarrow \beta$ holds.
- Augmentation rule : If $\propto \rightarrow \beta$, holds and γ is a set of attributes then $\gamma\propto \rightarrow \gamma\beta$ holds.
- Transitivity Rule: If $\propto \rightarrow \beta$ and $\beta \rightarrow \gamma\beta \rightarrow \gamma$ holds then $\propto \rightarrow \gamma$ holds

To simplify above rules there are another three rules.

- Union rule: If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ holds $\alpha \rightarrow \beta r$ holds.
- Decomposition Rule: If $\alpha \rightarrow \beta\gamma$ holds then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \beta$holds.
- Pseudo Transitivity Rule: If $\alpha \rightarrow \beta$ holds and $\beta\gamma$ ->δ holds then $\alpha\gamma \rightarrow \delta$ holds

Consider above example i.e. R (A, B, C, G, H, I) and set of functional dependencies (A→ B, A→C, CG→H, CG→I, B→H)

Find F^+ (Closure set of F using above given rules)

1. A→B, B→H ⇒A→H by transitivity property.
2. A→B, A→C ⇒ B→C by transitivity property.
3. i. A→C, CG→H ⇒ AG→H
   ii. A→C, CG→I ⇒ AG→I
   } By Pseudo Transitivity Rule
4. A→B, A→C => A→BC by union rule.
5. CG→H, CG→I=> CG→HI by union rule.

Example 2:

Computer F+ for the following Functional Dependency

R=(A,B,C,D)

F=(A→B, A→C, BC→D)

Solution :

F^+= A→BC	**Union** Rule A→B and A→C
A→D	Transitive Rule **A→BC ,**BC→D
A→BCD	Union Rule **A→BC,A→D**
AC→D	Pseudotrnasitivity Rule A→B, BC→D
AB→D	Pseudotrnasitivity Rule A→C, BC→D

Data Normalization:

Normalization is the formal process for deciding which attributes should be grouped together in relation.
Normalization is the systematic & scientific process for designing table.
Before processing with the physical design, validate and improve the logical design called normalization.
The process of normalization was first developed by E.F.Codd.
Normalization is the process of building database structure to store data.
Normalization is a formal process of developing data structures in a manner that which eliminates redundancy and promote integrity.
Normalize design would have the following properties.

1. Data storage requirement would be minimize, since the normalization process systematically removes duplication of data.
2. Since data items are stored in the minimum number of places, so data inconsistency would be minimize.
3. Normalized structure would be optimal for updates at the expenses of retrieval (Insert, Update and Delete).

A normal form is a state of table that results from applying simple rules regarding functional dependency or relationship between attributes.

Primary Key and Candidate Key

The column in the table which uniquely identify the record in the database called primary key. This column cannot take duplicate values some times more than one column is required to uniquely identify a row i.e. combination of these column forms a primary key. Each of these column is then known as a Candidate Key .
consider following example :
e.g. Drawing license no, customer code, pass port number and customer name.

1. None of the above column can take null value.
2. No two rows can have identical:-
    a. Drawing license_no.
    b. Customer code.
    c. Pass port no.

So above three attributes becomes a candidate for the primary key or candidate key. In such a case we can select one of them as a primary key then other two keys are called “Alternate Keys”

Foreign key/Reference/Secondary key:.

A foreign key is a column in table which uniquely identifies a record from a different table. So primary key and foreign key are interrelated in which foreign key is used to store values of another tables primary key to describes the relationship between data from different tables.

Types of Normal Form :
First Normal Form (1NF) : The multi-valued attributes (called repeating groups) should be removed i.e. elimination of repeating groups.
Second Normal Form (2NF) : The partial functional dependency have to be removed i.e. elimination of redundant data.
Third Normal Form (3NF) : The transitive dependencies have to be removed , i.e. eliminate columns not dependant on the key.
Boyce-Codd Normal Form (BCNF) : The remaining anomalies that result from functional dependencies are removed.

UnnormalisedRelation :

How to recognize that the relation in normalized? Before that we have to study Normalization we must not how to recognize the normalized and unnormalised relation.
If any relation contains repeating group of elements then that relation is called as normalized relation.

In following Inventory Table all the columns contain Data with some repeating values. In the definition of Normalisationit is not allowed to have repetitive data in the following table near about all columns contain of repetitive data.

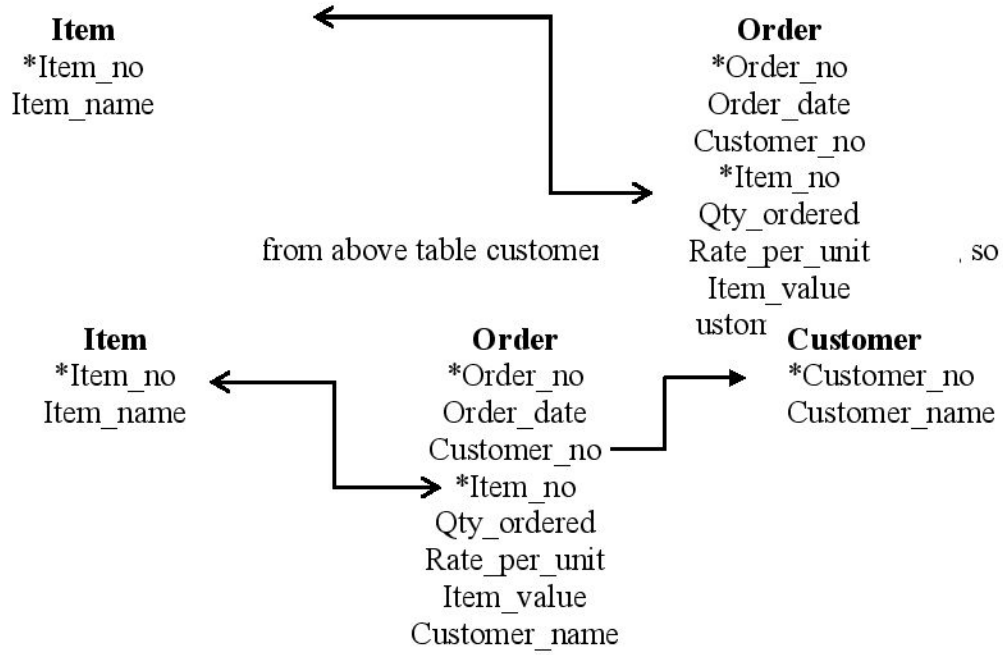
ORDERNO	ORDER_DATE	CUSTOMER_NO	CUSTOMER_NAME	ITEM_NO	ITEM_NAME	Qty	Rate	TOTAL
101	10/5/2012	1	RAJAN	10	CUPBOARD	2	2000	4000
101	10/5/2012	1	RAJAN	20	CHAIR	5	500	2500
102	11/5/2012	7	SHITAL	30	COMPUTER	2	35000	70000
102	11/5/2012	7	SHITAL	40	PRINTER	1	10000	10000
102	11/5/2012	7	SHITAL	10	CUPBOARD	1	1000	1000

Table 1: Unnormalised Data

First Normal Form:

Consider relation
item_order(orderno,order_date,customer_no,item_no,qty_ordered,rate_per_item,item_value,customer_name)
A relation is in the first normal form if it does not contain any repeating elements or groups.
It is the relation in which the intersection of each row and column contains one and only one value. We identify and remove repeating groups within a table.
Repeating group is a set of columns that stored similar information which is repeated in the same table.
A relation is in the first normal form only if all underlined domain contains only atomic value. Since our example contains the repeating groups of item details so it is not in the first normal form.

To convert a relation into the first normal form, remove all repeating or multi valued attributes to another child relation. So above table is now converted into first normal form in the following table.



Above tables ORDER,ITEM& CUSTOMER contains relative data items. Each of the data item functional dependent on Primary key of each table and also avoid repetitive type of data as.

ORDERNO	ORDER_DATE
101	10/5/2012
102	11/5/2012

ORDER TABLE

CUSTOMER_NO	CUSTOMER_NAME
1	RAJAN
7	SHITAL

CUSTOMER TABLE

ITEM_NO	ITEM_NAME	QTY_ORDERED	RATE_PER_ITEM	TOTAL
10	CUPBOARD	2	2000	4000
20	CHAIR	5	500	2500
30	COMPUTER	2	35000	70000
40	PRINTER	1	10000	10000
10	CUPBOARD	1	1000	1000

ITEM TABLE **Fig- 1NF**

Second Normal Form:

A relation in 2nd N.F. if it is in the 1st N.F. The following are the rules for relation is in the 2nd N.F.:-

1. The primary key consists of only one attribute.
2. No non-key attributes exist in the relation.
   i.e. all the attributes in the relation are components of primary key.
3. Every non-key attribute is functionally dependent on the full set of Primary Key attributes.

In above relation we find that above three relations are now in the 2nd N.F. i.e. we have assign primary key in every relation.

i.e. Order table has primary key order_no and item_no, all non-key attributes are depend on primary key i.e. order_no and item_no.

In item relation item_no is the primary key and item_name is dependent on item_no. We have also created 3rd relation customer which has primary key customer_no where customer_name is dependent on customer_no.

Third Normal Form:

1. A relation is in the 3rd N.F. if it is in the 2nd N.F.
2. There is no transitive dependencies exist.
3. Transitive Dependencies in a relation is for functional dependencies between two or more non-key attributes.
4. Column in each table should be group of columns in which the data in each column contributes to the description of each row in the table.
5. For a given row with a unique key, each column appearing in that row should contribute to the description of that row.

Now by following above rules, given relation is in 3rd N.F. which has the following changes.

Order	**Order_item**	**Customer**	**Item**
*Order_no	*order_no	*customer_no	*item_no
Order_date	item_no	customer_name	item_name
Customer_no	qty_order		
	Rate_per_unit		

Boyce Codd Normal Form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Consider aexample , following relation is in 3NF and we have to convert it into BCNF

Student_name	**Course_name**	**Teacher_name**
Vijay Patil	DBMS	Mrs.Radha
Shreyas Rana	Java Programming	Mr.Sumit
NayanApte	Big Data analytics	Rajan Kelkar
Sonali Joshi	Artificial Intelligence	Miss.Sanjana Patil
Shital Patil	Java Programming	Mr.Sumit
Radhika Chahel	Big Data analytics	Rajan Kelkar

In the above relation key is

Key - {Student_name,Course_name}, Teacher_name is not a super key

Here Student_name,Course_name form Primary key, means Course_name is Prime attriute.

But there is one more dependency Teacher_name→Course_name

And Course_name is prime attribute and Teacher_name is non-prime attribute and is not allowed in BCNF.

BCNF Process

In the above relation there some problems

1. If student Vijay Patil and Sonali Joshi left the course of cancel the admission record of their teacher and course will be automatically deleted .
2. Repetition of course- name and teacher- name

To avoid above problems we split above 3NF relation in BCNF

Student Relation

Student_name	**Course_name**
Vijay Patil	DBMS
Shreyas Rana	Java Programming
NayanApte	Big Data analytics
Sonali Joshi	Artificial Intelligence
Shital Patil	Java Programming
Radhika Chahel	Big Data analytics

Course Relation

Course_name	**Teacher_name**
DBMS	Mrs.Radha
Artificial Intelligence	Miss.Sanjana Patil
Java Programming	Mr.Sumit
Big Data analytics	Rajan Kelkar

In above both relations student and course there is no any problem remains and above relations are in BCNF

Comparison Chart

BASIS FOR COMPARISON	3NF	BCNF
Concept	No non-prime attribute must be transitively dependent on the Candidate key.	For any trivial dependency in a relation R say X->Y, X should be a super key of relation R.
Dependency	3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved in BCNF.
Decomposition	Lossless decomposition can be achieved in 3NF.	Lossless decomposition is hard to achieve in BCNF.

MCQ on Relational Data Model

1. In the _____ normal form, a composite attribute is converted to individual attributes.

A) First

B) Second

C) Third

D) Fourth

2. A table on the many side of a one to many or many to many relationship must:

a) Be in Second Normal Form (2NF)

b) Be in Third Normal Form (3NF)

c) Have a single attribute key

d) Have a composite key

Answer:d
Explanation: The relation in second normal form is also in first normal form and no partial dependencies on any column in primary key.

3. Tables in second normal form (2NF):

a) Eliminate all hidden dependencies

b) Eliminate the possibility of a insertion anomalies

c) Have a composite key

d) Have all non key fields depend on the whole primary key

Answer:a
Explanation: The relation in second normal form is also in first normal form and no partial dependencies on any column in primary key.

4. Which-one ofthe following statements about normal forms is FALSE?

a) BCNF is stricter than 3 NF

b) Lossless, dependency -preserving decomposition into 3 NF is always possible

c) Loss less, dependency – preserving decomposition into BCNF is always possible

d) Any relation with two attributes is BCNF

Answer:c
Explanation: We say that the decomposition is a lossless decomposition if there is no loss of information by replacing r (R) with two relation schemas r1(R1) andr2(R2).

5. Functional Dependencies are the types of constraints that are based on_____

a) Key

b) Key revisited

c) Superset key

d) None of these

Answer:a
Explanation: Key is the basic element needed for the constraints.

6. Which is a bottom-up approach to database design that design by examining the relationship between attributes:

a) Functional dependency

b) Database modeling

c) Normalization

d) Decomposition

Answer:c
Explanation: Normalisation is the process of removing redundancy and unwanted data.

7. Which forms simplifies and ensures that there is minimal data aggregates and repetitive groups:

a) 1NF

b) 2NF

c) 3NF

d) All of the mentioned

Answer:c
Explanation: The first normal form is used to eliminate the duplicate information.

8. Which forms has a relation that possesses data about an individual entity:

a) 2NF

b) 3NF

c) 4NF

d) 5NF

Answer:c
Explanation: A Table is in 4NF if and only if, for every one of its non-trivial multivalued dependencies X \twoheadrightarrow Y, X is a superkey—that is, X is either a candidate key or a superset thereof.

9. Which forms are based on the concept of functional dependency:

a) 1NF

b) 2NF

c) 3NF

d) 4NF

Answer:c
Explanation: The table is in 3NF if every non-prime attribute of R is non-transitively dependent (i.e. directly dependent) on every superkey of R.

10. Empdt1(empcode, name, street, city, state,pincode).

For any pincode, there is only one city and state. Also, for given street, city and state, there is just one pincode. In normalization terms, empdt1 is a relation in

a) 1 NF only

b) 2 NF and hence also in 1 NF

c) 3NF and hence also in 2NF and 1NF

d) BCNF and hence also in 3NF, 2NF and 1NF

Answer:b
Explanation: The relation in second normal form is also in first normal form and no partial dependencies on any column in primary key.

11. We can use the following three rules to find logically implied functional dependencies. This collection of rules is called

a) Axioms
b) Armstrong's axioms
c) Armstrong
d) Closure

Answer:b
Explanation:By applying these rules repeatedly, we can find all of F+, given F.

12. Which of the following is not a Armstrong's Axiom ?

a) Reflexivity rule

b) Transitivity rule

c) Pseudotransitivity rule

d) Augmentation rule

Answer:c
Explanation:It is possible to use Armstrong's axioms to prove that Pseudotransitivity rule is sound.

13. The relation employee(ID,name,street,Credit,street,city,salary) is decomposed into
employee1 (ID, name)
employee2 (name, street, city, salary)
This type of decomposition is called
a) Lossless decomposition
b) Lossless-join decomposition
c) Both a and b
d) None of the mentioned

Answer:d
Explanation:Lossy-join decomposition is the decomposition used here .
14. Inst_dept (ID, name, salary, dept name, building, budget) is decomposed into
instructor (ID, name, dept name, salary)
department (dept name, building, budget)
This comes under
a) Lossy-join decomposition
b) Lossy decomposition
c) Lossless-join decomposition
d) Both a and b

Answer:d
Explanation:Lossy-join decomposition is the decomposition used here .
15. There are two functional dependencies with the same set of attributes on the left side of the arrow:
A->BC
A->B
This can be combined as
a) A->BC
b) A->B
c) B->C
d) None of the mentioned

Answer:a
Explanation:This can be computed as the canonical cover .
16. Consider a relation R(A,B,C,D,E) with the following functional dependencies:
ABC -> DE and

D -> AB

The number of superkeys of R is:

a) 2

b) 7

c) 10

d) 12

Answer:c
Explanation:Asuperkey is a combination of columns that uniquely identifies any row within a relational database management system (RDBMS) table.

One or Two mark Questions :

1. What do you mean by data integrity?
2. What do you mean by entity integrity?
3. What do you mean by referential integrity?
4. What do you mean by the term integrity constraints?
5. What are different types of integrity constraints?
6. What is base table integrity constraints?
7. Define : Primary key, Foreign key, Alternate key, Candidate key
8. What is Normalisation?
9. What is referential integrity?

Brief Questions:

1. What is Normalisation? Explian the process up to 3NF
2. Explain Normalisation with suitable Example
3. What is Functional Dependency? How to compute closure of Functional Dependency?

4. Computer the Functional Dependency closure(F+)for the following

   R=(A,B,C,D)
   F=(A$\rightarrow$B, A$\rightarrow$C, BC$\rightarrow$D)

5. Computer the Functional Dependency closure(F+)for the following

   Suppose that the schema R = (A, B, C, G, H, I) and the set of function dependencies
   A $\rightarrow$ B
   A $\rightarrow$ C

$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H$

6. State and Explain any four codd's rule

7. Define Integrity constraint? Explain Referential Integrity with example

CHAPTER - 4. RELATIONAL ALGEBRA

4.1 Preliminaries

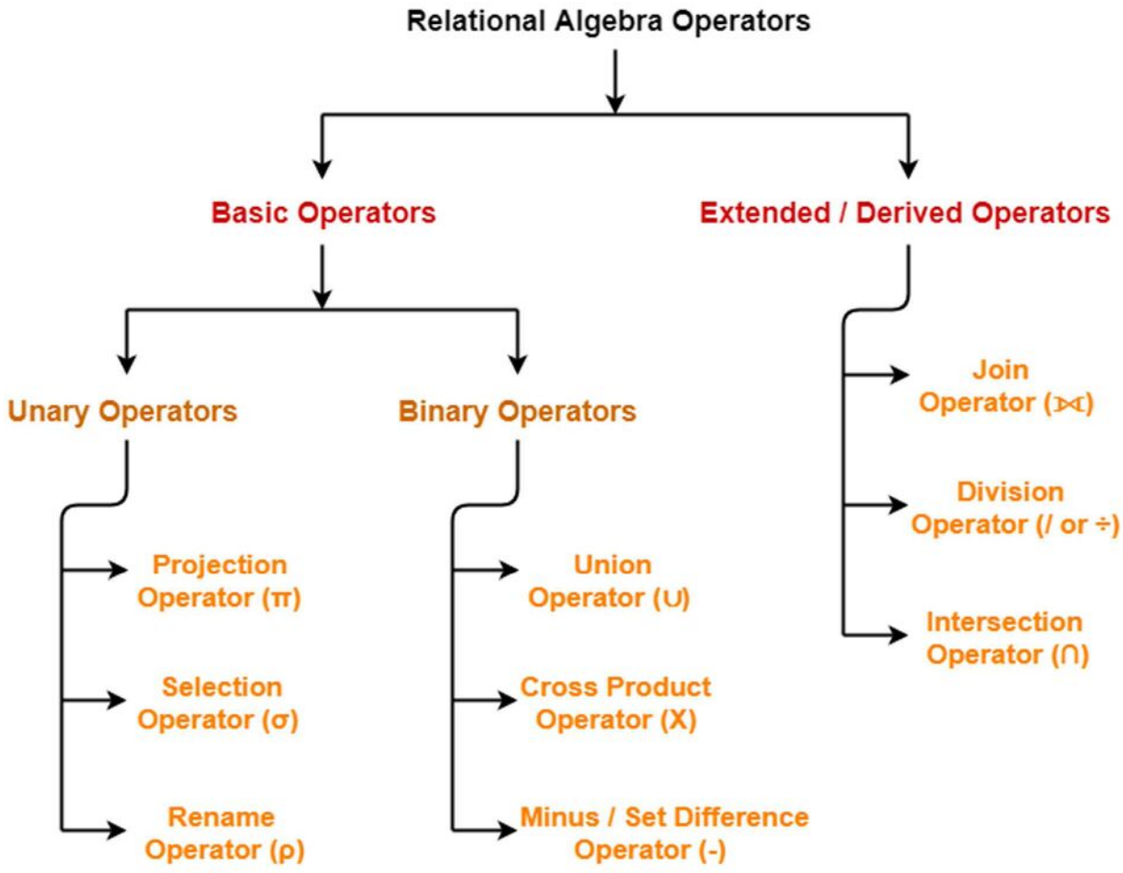
The basic set of operations for relational model is the Relational Algebra. These operations enable uses to retrieve several results. The Relational Algebra is is very important for several reasons.

i. It provides a formal foundation for relational model operations.
ii. Useful for implementing and optimising Queries for RDBMS.

Relational algebra is a **procedural query language**. It takes one or more relations / tables and performs the operation and produce the result. This result is also considered as a new table or relation.

Relational algebra will have operators to indicate the operations. This algebra can be applied on single relation – called **unary** or can be applied on two tables – called **binary**. While applying the operations on the relation, the resulting subset of relation is also known as new relation. There can be multiple steps involved in some of the operations. The subsets of relations at the intermediary level are also known as relation. We will understand it better when we see different operations below.

4.2 Relational Algebra :A small set of set-based operators that allow us tomanipulate relations in limited but useful ways



.

There are two categories of Relational Algebric operators.

Unary operators : Operators which operates on single relation are unary operators . Following are the unary operators.

Operation	Operator	Symbol
Projection	**PROJECT**	π
Selection	**SELECT**	σ
Renaming	**RENAME**	ρ

Binary Operators: Operators which operates on Two or Multiple relations are Binary operators. Following are the Binary operators.

Operation	**Operator**	Symbol
Union	**UNION**	$\cup$
Intersection	**INTERSECTION**	$\cap$
Cartesian product	**X**	$\times$
Join	**JOIN**	$\bowtie$

1. The Select Operation(σ)

Select selects tuples that satisfy a given predicate. Select is denoted by a lowercase Greek sigma (σ), with the predicate appearing as a subscript. The argument relation is given in parentheses following the σ.

For example, to select tuples (rows) of the *Student* relation where the Subject is Computer we would write

σ Subject='Computer'

Student

Rollno	Name	Class	Subject	Dtofadm	Fees paid
112	Rajshri Patil	FYBSC	Computer	10 Jun 2012	2340
592	SubhashRane	TYBSC	Physics	14 AUG 2012	2540
234	Pratik Patil	FYBSC	Computer	14 Jun 2012	1890
612	Rani Mehera	TYBSC	Electronics	12 Aug 2012	2340
114	Trinad Sharma	SYBSC	Computer	27 Jun 2012	1970

Rank

Rollno	Marks	Medal	Grade
112	600	Branz	C
592	800	Silver	B
234	832	Silver	B
612	952	Gold	A
114	435	--	D

Figure 3.3: The *Student* and *Rank* relations.

We allow comparisons using =, $\neq$, <, $\leq$, > and $\geq$in the selection predicate.

We also allow the logical connectives $\vee$(or) and $\wedge$(and). For example:

σ Subject='Computer' $\wedge$rollno>250

2. The Project Operation (π)

Project operations selects certain columns from relation. It does not reduce rows.

Projection is denoted by the Greek capital letter pi (**Π**). The attributes to be copied appear as subscripts.

Duplicate rows are automatically eliminated, as relation is a set.

We can perform these operations on the relations resulting from other operations.

Example :

- To get name of students from student relation

  π_{name} (Student)

- To get name and rollno of student who paid more than 2000 fess

$\pi_{name\ ,rollno}(\sigma_{fees_paid> 2000})^{(Student)}$

Think of **select** as taking rows of a relation, and **project** as taking columns of a relation.

Binary Operators

1. **The Union Operation**(∪)

The **union** operation is denoted ∪as in set theory. It returns the union (set union) of two compatible relations.

For a union operation $r \cup s$to be legal, we require that

2. $r$and$s$must have the same number of attributes.
3. The domains of the corresponding attributes must be the same.

Example :

Sport_club

Employee name	Dept. Name
Ram Patil	Finance
Shital Tiwari	Computer
Ranjana Pathak	HR
Suresh Pillai	Computer
Sanjay Bagul	Finance
Manoj Bajpai	Computer
Ratan Rajput	IT
Archana Deshmukh	Finance
ManavPande	HR

Tea_club

Employee name	**Dept. Name**
Ram Patil	Finance
Sunder chawala	Computer
Sanjay Bagul	Finance
Hemant Bonde	Computer
Ratan Rajput	IT
MayurRathi	HR

As in all set operations, duplicates are eliminated, giving the relation of Figure

Result of Sport _club ∪Tea club

Employee name	**Dept. Name**
Ram Patil	Finance
Shital Tiwari	Computer
Ranjana Pathak	HR
Suresh Pillai	Computer
Sanjay Bagul	Finance
Manoj Bajpai	Computer
Ratan Rajput	IT
Archana Deshmukh	Finance
ManavPande	HR
Sunder chawala	Computer
Hemant Bonde	Computer
MayurRathi	HR

2. **The Intersection Operation** (∩)

The **Intersection** operation is denoted∩as in set theory. It returns common rows from f two compatible relations.

For a union operation ∩ to be legal, we require that

1. r and s must have the same number of attributes.
2. The domains of the corresponding attributes must be the same.

Example :

Result of Sport ∩ _club Tea club

Employee name	**Dept. Name**
Ram Patil	Finance
Sanjay Bagul	Finance
Ratan Rajput	IT

3. The Set Difference Operation

Set difference is denoted by the minus sign (_). It finds tuples that are in one relation, but not in another.

Let R and S be two relations.

Then-

- R – S is the set of all tuples belonging to R and not to S.
- In R – S, duplicates are automatically removed.
- Difference operation is associative but not commutative.

Example

Sport_club (R)

Employee name	Dept. Name
Ram Patil	Finance
Shital Tiwari	Computer
Ranjana Pathak	HR
Suresh Pillai	Computer
Manoj Bajpai	Computer
Archana Deshmukh	Finance
ManavPande	HR

Sport_club(S)

Employee name	Dept. Name
Ranjana Pathak	HR
Hemant Bonde	Computer
Manoj Bajpai	Computer

Result of R-S is

Employee name	Dept. Name
Ram Patil	Finance
Shital Tiwari	Computer
Suresh Pillai	Computer
Archana Deshmukh	Finance
ManavPande	HR

4. The Cartesian Product Operation (×)

The **cartesian product** of two relations is denoted by a cross (×), written

$$r_1 \times r_2 \text{ for relations } r_1 \text{ and } r_2$$

The result of $r_1 \times r_2$ is a new relation with a tuple for each possible **pairing** of tuples from r_1 and r_2.

In order to avoid ambiguity, the attribute names have attached to them the name of the relation from which they came. If no ambiguity will result, we drop the relation name.

Example :

Employee Salary

	Emp_Name
100	RajshkharPatil
101	NayanaBhole
102	ShradhaRane

Emp_no	Salary
100	45633
102	58437

Result of Employee X Salary

Emp_no	Emp_Name	Emp_no	Salary
100	RajshkharPatil	100	45633
100	RajshkharPatil	102	58437
101	NayanaBhole	100	45633
101	NayanaBhole	102	58437
102	ShradhaRane	100	45633
102	ShradhaRane	102	58437

5. Join Operation : (⋈)

The Join Operation combines two relations to form a new relation. The meaning of a join operation in terms of catesian production and selection operation is given below.
Result of Cartesian product of two relation is ambiguous as per above fig. After Selecting common attribute from both relation it generates correct result as

Example :
Consider above relations Employee and salary and join both relation as
Employee ⋈Salary which results only those rows which attribute Emp_no is common in both relations.

Emp_no	Emp_Name	Emp_no	Salary
100	RajshkharPatil	100	45633
102	ShradhaRane	102	58437

Employee ⋈ Salary

Example : To get common employees which present in both relations
σEmployee.Emp_no=Salary.Emp_no (Employee ⋈ Salary)

Division(/ or %)

- Notation: r / s or $r \div s$
- Useful for expressing queries that include a "**for all**" or "**for every**" phrase
- Let r and s be relations on schemas R and S respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  Then r/s is a relation on schema
  $R - S = (A_1, \ldots, A_m)$
  defined as
  $$r / s = \{ t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s (tu \in r) \}$$
- Informally, r / s contains the (parts of) tuples of r that are associated with every tuple in s.

Example : of Division A/B

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

A/B1

sno
s1
s2
s3
s4

A/B2

sno
s1
s4

A/B3

sno
s1

Example : Following are the relational Algebraic Queries based on Emp and Depart Tables

Emp

Dept	**Dname**	**Location**
CS	Computer	Pali hill
PY	Physics	Niral
Econ	Econimics	Pali hill

Depart

Name	**Dept**	**Salary**	**Rank**
Rajan	CS	34567	Assitant
Subha	PY	50000	Manager
Pankaj	CS	52000	Clerk
Chhya	Econ	12000	Assistant

1. Select only those Employees in the CS department:

   σ Dept = 'CS' (EMP)
2. Select only those Employees with last name Smith who are assistant professors:

   σ Name = 'Rajan' $\wedge$Rank = 'Assistant' (EMP)
3. Select only those Employees who are either Assistant Professors or in the Economics department:

   σ Rank = 'Assistant' $\vee$ Dept = 'Econ' (EMP)
4. Select only those Employees who are not in the CS department or Adjuncts:

   σ $\neg$ (Rank = 'Adjunct' $\vee$ Dept = 'CS') (EMP)
5. Show the names of all employees working in the CS department:

   π name σ (Dept = 'CS' (EMP))
6. Show the name and rank of those Employees who are not in the CS department or Adjuncts:

   π name, rank σ ($\neg$ (Rank = 'Adjunct' $\vee$ Dept = 'CS') (EMP))
7. Find all information on every employee including their department info:

   EMP emp.Dept = depart.Dept DEPART
8. Show the names of Dept in which more than 50000 salaried employee works

   πDept.Dname(σemp.Dept = depart.Dept$\wedge$Emp.salary>50000) $^{(Emp \bowtie Dept)}$

OPERATION	. PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation *R*	$\sigma_{<\text{selection condition}>}(R)$
PROJECT	Produces a new relation with only some of the attributes of *R*, and removes duplicate tuples.	$\pi_{<\text{attribute list}>}(R)$
THETA JOIN	Produces all combinations of tuples from *R*1 and *R*2 that satisfy the join condition.	$R1 \bowtie_{<\text{join condition}>} R2$
EQUIJOIN	Produces all the combinations of tuples from *R*1 and *R*2 that satisfy a join condition with only equality comparisons.	$R1 \bowtie_{<\text{join condition}>} R2$, OR $R \bowtie 1_{(<\text{join attributes 1}>),(<\text{join attributes 2}>)} R2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of *R*2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R1 *_{<\text{join condition}>} R2$, OR $R1 *_{(<\text{join attributes 1}>),(<\text{join attributes 2}>)} R2$ OR $R1 * R2$
UNION	Produces a relation that includes all the tuples in *R*1 or *R*2 or both *R*1 and *R*2; *R*1 and *R*2 must be union compatible.	$R1 \cup R2$
INTERSECTION	Produces a relation that includes all the tuples in both *R*1 and *R*2; *R*1 and *R*2 must be union compatible.	$R1 \cap R2$
DIFFERENCE	Produces a relation that includes all the tuples in *R*1 that are not in *R*2; *R*1 and *R*2 must be union compatible.	$R1 - R2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of *R*1 and *R*2 and includes as tuples all possible combinations of tuples from *R*1 and *R*2.	$R1 \times R2$

5. Database Implementation:

5.1 Database Security:

Security is an important issue in database management because information stored in a database is very valuable and many time, very sensitive materials. So the data in a database management system need to be protected from abuse and should be protected from unauthorized access and updates.

Database security is the protection of the database against intentional and unintentional threats that may be computer-based or non-computer-based. Database security is the business of the entire organization as all people use the data held in the organization's database and any loss or corruption to data would affect the day-to-day operation of the organization and the performance of the people. Therefore, database security encompasses hardware, software, infrastructure, people and data of the organization.

Data Security Risks

The organization should identify all the risk factors and weak elements from the database security Perspective and find solutions to counter and neutralize each such threat.

A threat is any situation, event or personnel that will adversely affect the database security and the smooth and efficient functioning of the organization. A threat may be caused by a situation or event involving a person, action or circumstance that is likely to bring harm to the organization. The harm may be tangible, such as loss of data, damage to hardware, loss of software or intangible such as loss of customer goodwill or credibility and so on.

Data Tampering

Privacy of communications is essential to ensure that data cannot be modified or viewed in transportation. The chances of data tampering are high in case of distributed environments as data moves between sites. In a data modification attack, an unauthorized party on the network intercepts data in transit and changes that data before retransmitting it. An example of this is changing the amount of a banking transaction from Rs. 1000 to Rs. 10000.

Data Theft

Data must be stored and transmitted securely, so that information such as credit card numbers cannot be stolen. Over the Internet and Wide Area Network (WAN) environments, both public carriers and private network owners often route portions of their network through insecure landlines, LAN cables, Satellite links etc.

Falsifying User Identities

In a distributed environment, it becomes more feasible for a user to falsify an identity to gain access to sensitive and important information. Criminals attempt to steal users' credit card numbers, and then make purchases against the accounts. Or they steal other personal data, such as bank account numbers and driver's license numbers, and make bogus credit accounts in someone else's name.

Password-Related Threats
In large systems, users must remember multiple passwords for the different applications and services that they use. Users typically respond to the problem of managing multiple passwords in several ways:

• They may select easy-to-guess password, such as name or a famous name may be easily detected by attacker

• They may also choose to standardize passwords so that they are the same on all machines or websites. By making slight variations in password that can be easily derived by known passwords

All these strategies compromise password secrecy and service availability. Moreover, administration of multiple user accounts and passwords is complex, time-consuming, and expensive.

Unauthorized Access to Tables and Columns
The database may contain confidential tables, or confidential columns in a table, which should not be available indiscriminately to all users authorized to access the database. It should be possible to protect data on a column level.

Unauthorized Access to Data Rows
Certain data rows may contain confidential information that should not be available indiscriminately to users authorized to access the table. For example, in a shared environment' businesses should have access only to their own data; customers should be able to see only their own orders.

Lack of Accountability
If the system administrator is unable to track users' activities, then users cannot be held responsible for their actions. There must be some reliable ways to monitor who is performing what operations on the data.

Complex User Management Requirements
The system must often support a large number of users and therefore they must be scalable.

In such large-scale environments, the burden of managing user accounts and passwords makes your system vulnerable to error and attack.

Data Security Requirements

We should use technology to ensure a secure computing environment for the organization. Although it is not possible to find a technological solution for all problems, most of the security issues could be resolved using appropriate technology. The basic security standards which technology can ensure are confidentiality, integrity, and availability.

1. **Confidentiality**

   A secure system ensures the confidentiality of data. This means that it allows individuals to see only the data they are supposed to see. Confidentiality has several aspects like privacy of communications, secure storage of sensitive data, authenticated users and authorization of users.

2. **Privacy of Communications**

   The DBMS should be capable of controlling the spread of confidential personal information such as health, employment, and credit records. It should also keep the corporate data such as trade secrets, proprietary information about products and processes, competitive analyses, as well as marketing and sales plans secure and away from the unauthorized people.

3. **Secure Storage of Sensitive Data**

   Once confidential data has been entered, its integrity and privacy must be protected on the databases and servers wherein it Resides.

4. **Authenticated Users**

   Organization should identify the authenticated user of the system .A user can respond to a request to authenticate by providing a proof of identity, or an authentication token

5. **Authorization**

   An authenticated user goes through the second layer of security, authorization. Authorization is the process through which system obtains information about the authenticated user, including which database operations that user may perform and which data objects that user may access.

   A user may have several forms of authorization on parts of the database. There are the following authorization rights.

   - Read authorization allows reading, but not modification, of data.
   - Insert authorization allows insertion of new data, but not modification of existing data.
   - Update authorization allows modification, but not deletion of data.
   - Delete authorization allows deletion of data.

6. Integrity

   A secure system en sums that the data it contains is valid. Data integrate means that data is protected from deletion and corruption, both while it resides within the data-case, and while it is being transmitted over the network.

7. A secure system makes data available to authorized users, without delay. Denial of service attacks are attempts to block authorized users' ability to access and use the system when needed.

5.2 Database Integrity

The preservation of the integrity of a database system is concerned with the maintenance of the correctness and consistency of the data In a multi-user database environment this is a major task, since integrity violations may arise from many different sources, such as typing errors by data entry clerks, logical errors in application programs, or errors in system software which result in data corruption.

Integrity rules may be divided into three 'broad categories:

- Domain integrity
- Entity integrity
- Referential integrity

5.3 Transaction Management

What is transaction?

A collection of several operation perform on the database is considered to be a single unit called transaction. Transaction is initiated by a user program written in high level programming languages (like C,C++,COBOL,PASCAL etc) . The transaction is enclosed between begin transaction and end transaction.

e.g.

1. To see or check account balance
2. Pay telephone bill
3. Pay water charges
4. Pay electricity bill
5. Register for Demand Draft or checkbook

What is transaction management?

Transaction management is the ability of a database management system to manage the various transactions that occur in within the system.

A database management system should ensure that the transactions are executed properly. A transaction is a unit of database activities that accesses and possibly updates various data items. These operations are performed using high level data manipulation language statements like beginning transaction, end transactions ,read ,write etc.

Transaction Concept

Transaction processing is a style of computing, typically performed by large server computers, that supports interactive applications. In transaction processing, work is divided into individual, indivisible operations, called transactions. By contrast, batch processing is a style of computing in which one or more programs processes a series of records (a batch) with little or no action from the user or operator.

A transaction processing system allows application programmers to concentrate on writing code that supports the business, by shielding application programs from the details of transaction management:

- It manages the concurrent processing of transactions.
- It enables the sharing of data.
- It ensures the integrity of data
- It manages the prioritization of transaction execution.

5.4 Transaction State:

A transaction must be in one of the following state

- **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state or any transaction is nearby complete.
- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts −
    - Re-start the transaction
    - Kill the transaction
- **Committed** − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

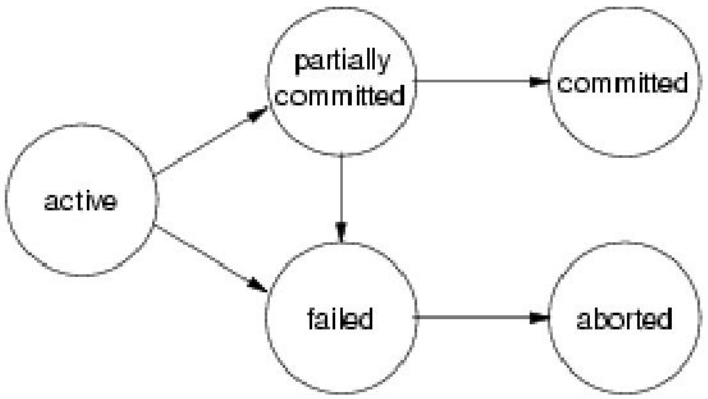


Fig : Transaction State

5.5 Transaction Properties:

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, andDurability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

- **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

- **Consistency** − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

- **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

- **Isolation** − In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

  Consider banking system e.g. which contains several accounts & a set of transactions that accesses & updates the account. Access to a database in unaccomplished by two operations given bellow
  Data is accessed by two basic operations.
  1. Read(x) : It reads the database item X into a program variable X
     This operation works as follows
     - Reading the disk block containing X into memory buffer
     - Transferring value of X from memory buffer to program variable X
  2. Write(x) : Writes the value of program variable X to database item X
     This operation works as follows
     - Reading the disk block containing X into memory buffer
     - Transferring value of program variable X to memory buffer. Storing the updated block from buffer back to disk.

Some sample Transactions are

T1	T2
Read(X)	Read(X)
X=X+100	Read(Y)
Write(X)	X=X+100
	Y=Y-50
	Write(X)
	Write(Y)

Fig : T1 & T2 sample Transactions

Database Structure:

We say a transaction has committed only after it has committed only after it has entered the committed state. Similarly we say that transaction aborted only after it has entered the aborted state. A transaction that is either committed or aborted is set to be terminated. A transaction contains a group of statements, which forms a logical unit of work. A transaction starts in the active state. When the transaction has finished executing the last statement it enters the partially committed state.

At this point the transaction has completed execution, but it is still possible that it may be aborted because of hardware failure. When the last of this information is written, the transaction enters the committed state. When transaction enters into the failed state due to hardware failure or logical error such transactions are rolled back.

When the roll back is compiled transaction enters into the aborted state. When a transaction aborts, the system has two options.

(1) Restarting Transaction. (2) Killed Transaction.

Questions :

1. What is Database Security?
2. What is Database Integrity?
3. Define Transaction
4. What is Data Tampering?
5. What is Data Theft?
6. State Transaction states
7. State ACID properties
8. What is Atomicity?
9. What is Consistency?
10. What is durability?
11. What is Isolation?

Broad Questions

1. What is Database Security? Explain any two Data security risks
2. State and Explain Data security requirements

3. What is Transaction? Explain Transaction states with suitable diagram.
4. Explain ACID properties
5. Explain the concept of transaction
6. What is Database Integrity? Explain Domain Integrity

MCQ on Transaction Management

1. Consider money is transferred from (1)account-A to account-B and (2) account-B to account-A. Which of the following form a transaction ?
a) Only 1
b) Only 2
c) Both 1 and 2 individually
d) Either 1 or 2

Answer:c
Explanation:The term transaction refers to a collection of operations that form a single logical unit of work.

2. A transaction is delimited by statements (or function calls) of the form _____.

a) Begin transaction and end transaction

b) Start transaction and stop transaction

c) Get transaction and post transaction

d) Read transaction and write transaction

Answer:a
Explanation:The transaction consists of all operations executed between the begin transaction and end transaction.

3. Identify the characteristics of transactions

a) Atomicity

b) Durability

c) Isolation

d) All of the mentioned

Answer:d
Explanation:Because of the above three properties, transactions are an ideal way of structuring interaction with a database.

4. Which of the following has “all-or-none” property ?

a) Atomicity

b) Durability

c) Isolation

d) All of the mentioned

Answer:a
Explanation:Either all operations of the transaction are reflected properly in the database, or none are.

5. The database system must take special actions to ensure that transactions operate properly without interference from concurrently executing database statements. This property is referred to as

a) Atomicity

b) Durability

c) Isolation

d) All of the mentioned

Answer:c
Explanation:Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions Ti and Tj , it appears to Ti that either Tj finished execution before Ti started or Tj started execution after Ti finished.

6. The property of transaction that persists all the crashes is

a) Atomicity

b) Durability

c) Isolation

d) All of the mentioned

Answer:b
Explanation:After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

7. _____ states that only valid data will be written to the database.

a) Consistency

b) Atomicity

c) Durability

d) Isolation

Answer:a
Explanation:If, for some reason, a transaction is executed that violates the database's consistency rules, the entire transaction will be rolled back and the database will be restored to a state consistent with those rules.

8. ____ means that the data used during the execution of a transaction cannot be used by a second transaction until the first one is completed.

a) Consistency

b) Atomicity

c) Durability

d) Isolation

Answer:d
Explanation:Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions Ti and Tj , it appears to Ti that either Tj finished execution before Ti started or Tj started execution after Ti finished.

6. Concurrency Control, Backup and Recovery

Concurrency Control:

Concurrency Control is the process of managing simultaneous operations like queries, updates, insert, delete, etc. on the database without having then interfered with one another. We know that most important property of the data stored in the database is it can be shared among different users and applications. So, important advantages access shared data concurrently.

If all the users of the database where only reading the data then concurrent access are very easy i.e. nobody is making any change to the data in the database.

But, if two or more users accessing the database simultaneously and if at least one user is updating the data there can be interferes, which results inconsistency.

To objective of concurrency control is like a multi-user computer system like UNIX or MVS & multi-user application programs like a banking system, railway reservation etc, Where many users can perform different operations at the same time.

Multiple Update Problems:

When the transactions are executed one after the other (serially), database always remains in consistent state. In other words, the effect created by multiple transactions is correct.

Concurrent transactions may cause inconsistency. This can be understood by following example.

Transaction T1	Transaction T2	Value of A in Database
Read(A)		500
	Read(A)	
A=A+100		
	A=A-100	
Write(A)		600
	Write(A)	400

Execution of Transaction

How above schedule generates wrong result

- Both T1 and T2 read initial value of A is 500
- In T1 add 100 to A then A becomes 600
- In T2 substract 100 from A becomes 400, this is the final value of A hence changes made in T1 are lost
- Generates wrong result hence not acceptable

In the above example both T1 and T2 executed concurrently. Write (A) operation in T1 performed before Write (A) operation in T2.

After execution value of A becomes 600 and 400 which is wrong, hence result is not acceptable.

Serializability:

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transaction are interleaved with some other transaction.

Schedule − A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.

Serial Schedule − It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

The objective of concurrency control is to arrange the transactions in such a way that to avoid any interferes, solving of interferes of the transaction is to execute then one at a time i.e. one transaction is committed before another is allowed to it is called serial schedule. In a serial schedule the transactions are performed in serial order. There is no interferes between the transactions since only one transaction is being executed at any given time. The order of execution can be important in a serial execution depending on the nature of the transactions involved.

A non serial schedule is a schedule where the operations from a group of concurrent transactions are involved. The objective of Serializability is to find non serial schedule which allows transactions to execute concurrently without interfering with one another. Such a schedule is said to be serializable.

To prevent inconsistency in concurrent processing of transactions the order of the read and write operation are important.

Serializability is an important property for concurrent transaction running in interleaved fashion.Interleaved execution of set of transaction is said to be serializable if it produces the same result of some serial execution of the transaction.

Example 1:

T1	T2
Read(A) A=A-100 Write(A) Read(B) B=B+100 Write(B)	
	Read(A) A=A+200 Write(A)

Fig (a) - Serial Schedule 1

T1	T2
	Read(A) A=A+200 Write(A)
Read(A)	

A=A-100 Write(A) Read(B) B=B+100 Write(B)	

Fig (b) - Serial Schedule 2

For serial schedule there should be the possibilities that all transaction of T2 executes after Transaction of T1 in fig(a) or Transaction of T1 executes after transaction of T2 in fig(b).

From above serial schedule we conclude that serial schedule generates correct result as

If Initial values of A=500 and B=1000 the schedule fig(a) and fig(b) gives
A=600 and b=1100

T1	T2
Read(A) A=A-100	
	Read(A) A=A+200
Write(A) Read(B)	
	Write(A)
B=B+100 Write(B)	

Non Serial schedule (c)

Non serial schedule (c) generates incorrect values of A=700 and B=1100

Some Non – Serial schedule generates correct values as serial schedule such schedules are called as Serializable.

Following Non serial schedule generates correct result as a=600 and b=1100 as serial schedule generates.

T1	T2
Read(A) A=A-100 Write(A)	
	Read(A) A=A+200 Write(A)
Read(B) B=B+100 Write(B)	

Non serial Schedule (d)

In the concurrent execution of transaction, it is very difficult to perform operation on transaction about their interaction. Instead we consider only two operations read and Write ,

transaction may perform arbitrary sequence of operation on data item Q . Only Read and Write operations are very important. We shall therefore show both operations in schedules as in fig (e).

T1	T2
Read(A) Write(A)	
	Read(A) Write(A)
Read(B) Write(B)	
	Read(B) Write(B)

fig (e).

Concurrency control schemes:
Process of managing simultaneous execution of transactions in a shared database, to ensure the Serializability of transactions, is known as concurrency control.
Simultaneous execution of transactions over a shared database can create several data integrity and consistency problems:

- Lost Updates.
- Uncommitted Data.
- Inconsistent retrievals.

Need of Concurrency Control
Concurrent access to data is desirable when:

1. The amount of data is sufficiently great that at any given time only fraction of the data can be in primary memory & rest should be swapped from secondary memory as needed.
2. Even if the entire database can be present in primary memory, there may be multiple processes.

Two of the most popular protocol which insures serializable results is locking and time stamping

6.1 Lock Based Protocols:

Locking is a procedure used to control concurrent access to the data when one transaction is accessing the database, a lock may deny access to other transactions to prevent incorrect results.

The most common method used to implement this requirement is to allow a transaction to access a data item only if it is currently holding a lock on that item.

✸ Locks:

There are various modes in which a data items may be locked. In this section we restrict our attention of two modes.

1. Shared

2. Exclusive

1. Shared:

If a transaction Ti has obtains a shared mode locks denoted by on item Q. then Ti can read but cannot write Q.

2. Exclusive:

If a transaction Ti has obtain an exclusive mode lock (denoted by x) on item Q, then Ti can both read and write Q.

Note that shared mode is compatible with shared mode but not with exclusive mode. At any time several shared mode locks can be hold simultaneously (by different transaction) on a particular data item a subsequent exclusive mode lock request has to wait until the currently held shared mode locks are released.

A transaction requests a shared lock on data item Q by executing the lock S (Q) instruction. Similarly, an exclusive lock is requested through the lock –x (Q) instruction a data item Q can be unlocked via the unlock Q instruction.

To access a data item transaction Ti must first lock that item. In the data item is already locked by another transaction in an incompatible mode, the concurrency control manager will not grant the lock until all incompatible locks held by other transaction have been released. ThusTi is made to wait until all incompatible locks held by other transaction have been released.

Transaction Ti may unlock a data item that it has locked at some earlier point. Note that a transaction must hold a lock on data item as long a transaction to unlock a data item immediately after its final access of that data item is not always desirable since serialization may not be ensured.

	S	X
S	True	False
X	False	False

Figure: Lock Compatibility matrix computer

Let A and B be two accounts that are accessed by transaction T1 and T2 transaction T1 transfer $50 from account B to account A and is defined as.

```
T1:   lock –x (B);
      Read (B);
      B: =B-50;
      Write (B);
      Unlock (B);
      Lock –X (A);
      Read (A);
      A: =A+50;
      Write (A);
      Unlock (A);
```

Transaction T2 displays the total amount of money in accounts A and B that is the sum A+B and defined as

```
T2:   lock –S(A);
      Read (A);
      Unlock (A);
```

Lock –S(B);
Read (B);
Unlock (A);
Display (A+B);

Suppose that the values of account A and B are Rs.100 and Rs.200 respectively. If these two transactions are executed serially, either in the order T1, T2 or the order T2, T1 then transaction T2 will display the value 300. If however, there transactions are executed concurrently, then schedule 1 as shown in the Following figureis possible. In this case transaction T2 will display 250, which is incorrect. The reason for this mistake is that the transaction T1 unlocked data item B too early, as a result of which T2 saw an inconsistent state.

T1	T2
lock –X (B) Read (B) B: =B-50 Write (B); Unlock (B)	
	lock –S(A) Read (A) Unlock (A) Lock –S(B) Read (B) Unlock (A) Display (A+B)
Lock –X (A) Read (A) A: =A+50 Write (A) Unlock (A)	

Concurrent Execution of T1 and T2

Pitfalls of Lock-Based Protocols

- Consider the partial schedule
- Neither T3 nor T4 can make progress — executing lock-S(B) causes T4 to wait for T3 to release its lock on B, while executing lock-X(A) causes T3 to wait for T4 to release its lock on A.
- Such a situation is called a deadlock.
- To handle a deadlock one of T3 or T4 must be rolled back and its locks released.

These data items are then available to the other transaction which can continue with its execution. If we do not use locking or unlock data items as soon as possible after reading or writing them we may get inconsistent states. On the other hand requesting a lock on another data item deadlock may occur. There are way to avoid deadlock in some situation locking is useful. Deadlocks are definitely preferable to inconsistent state, since they can be handled by rolling back of transactions where as inconsistent state may lead to real wand problem that cannot be handled by the database system.

We shall required that each transaction in the system follow a asset of rules called a locking protocol, indicating when a transaction may lock and unlock each of the data items locking protocols each of the data items locking protocols restrict the number of schedules is a proper subset of all possible serialization schedules.

Two Phase Locking Protocol

Once protocol that ensures serialization is the two phase locking protocol. This protocol requires that each transaction issue lock and unlock in two phases.

1. **Growing Phase**

   A transaction may obtain locks but may not release any lock.

2. **Shrinking Phase**

   A transaction may release locks, but may not obtain any new locks. Initially, a transaction is in the growing phase. The transaction a requires locks as needed. Once the transaction release a lock it enters the shrinking phase and it can issue no more lock requests.

Transaction T3 & T4 are two phase on the other hand transaction T1 & T2 are not two phase. Note that the unlock instructions do not need to appear at the end of the in the case of transaction T3 we would move unlock (B) instruction to just after the lock –x (A) instruction, and still retain the two phase locking property.

```
T3:   Lock –x (B);
      Read (B);
      B: = B-50;
      Write (B);
      Lock –x (A);
      Read (A);
      A: = A+50;
      Write (A);
      Unlock (B);
```

```
        Unlock (A);
T4:     Lock –s (a);
        Read (A);
        Lock –s (B);
        Read (B);
        Display (A+B);
        Unlock (A);
        Unlock (B);
```

Figure: Two phase locking.

5.2 Timestamp-Based Protocols

The locking protocols that we have described thus far determine the order between every pair of conflicting transactions at execution time by the first lock that both members of the pair request that involves incompatible modes. Another method for determining the serializability order is to select an ordering among transactions in advance. The most common method for doing so is to use a *timestamp-ordering* scheme.

Timestamps

With each transaction Ti in the system, we associate a unique fixed timestamp, de- noted by TS(Ti). This timestamp is assigned by the database system before the trans- action Ti starts execution. If a transaction Ti has been assigned timestamp TS(Ti), and a new transaction Tj enters the system, then TS(Ti) < TS(Tj). There are two simple methods for implementing this scheme:

1. Use the value of the *system clock* as the timestamp; that is, a transaction's time- stamp is equal to the value of the clock when the transaction enters the system.

2. Use a **logical counter** that is incremented after a new timestamp has been assigned; that is, a transaction's timestamp is equal to the value of the counter when the transaction enters the system.

The timestamps of the transactions determine the serializability order. Thus, if TS(Ti) < TS(Tj), then the system must ensure that the produced schedule is equivalent to a serial schedule in which transaction Ti appears before transaction Tj .

To implement this scheme, we associate with each data item Q two timestamp values:

• **W-timestamp**(Q) denotes the largest timestamp of any transaction that executed write(Q) successfully.

• **R-timestamp**(Q) denotes the largest timestamp of any transaction that executed read(Q) successfully.

These timestamps are updated whenever a new read(Q) or write(Q) instruction is executed.

The Timestamp-Ordering Protocol

The **timestamp-ordering protocol** ensures that any conflicting read and write operations are executed in timestamp order. This protocol operates as follows:

1. Suppose that transaction Ti issues read(Q).
a. If TS(Ti) < W-timestamp(Q), then Ti needs to read a value of Q that was already overwritten. Hence, the read operation is rejected, and Ti is rolled back.

b. If TS(Ti) ≥ W-timestamp(Q), then the read operation is executed, and R- timestamp(Q) is set to the maximum of R-timestamp(Q) and TS(Ti).

2. Suppose that transaction Ti issues write(Q).
a. If TS(Ti) < R-timestamp(Q), then the value of Q that Ti is producing was needed previously, and the system assumed that that value would never be produced. Hence, the system rejects the write operation and rolls Ti back.

b. If TS(Ti) < W-timestamp(Q), then Ti is attempting to write an obsolete value of Q. Hence, the system rejects this write operation and rolls Ti back.

c. Otherwise, the system executes the write operation and sets W-time- stamp(Q) to TS(Ti).

If a transaction Ti is rolled back by the concurrency-control scheme as result of issuance of either a read or write operation, the system assigns it a new timestamp and restarts it.

To illustrate this protocol, we consider transactions $T14$ and $T15$. Transaction $T14$ displays the contents of accounts A and B:

T_{14}: read(B);
read(A);
display($A + B$).

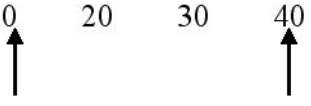
Transaction T_{15} transfers $50 from account A to account B, and then displays the contents of both:

T_{15}: read(B);
$B := B - 50$;
write(B);
read(A);
$A := A + 50$;
write(A);
display($A + B$).

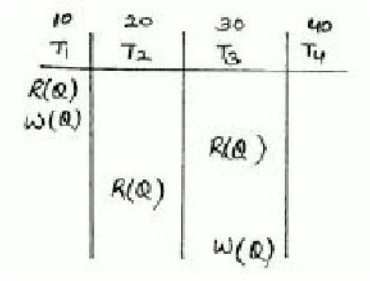
Example :

Consider four transactions executed as with their timestamps

T1	T2	T3	T4
10	20	30	40



Oldest Trn youngest trn



Let initially Q have no value i.e. 0. i.e.

RTimestamp=0 or RTS(Q) = 0 and

WTimestamp= 0 or WTS(Q) = 0

RTS(Q) = 0 and WTS(Q) = 0 means that no transaction has read or write the data item Q.

When Transaction issues Read(Q) :

Suppose T1 executes Read(Q) whose TS(T1) = 10, then we will have

RTS(Q) = ~~0~~ 10.

Then, T3 executes Read(Q) whose TS(T3) = 30, then the value of

RTS(Q) = ~~10~~ 30.

Now, if T2 executes Read(Q) whose TS(T2) = 20, then the value of

RTS(Q) will remain 30.

When Transaction issues Write(Q) :

Suppose T1 executes Write(Q) whose TS(T1) = 10, then we will have

WTS(Q) = ~~0~~ 10

Then, T4 executes Write(Q) whose TS(T4) = 40, then the value of

WTS(Q) = ~~10~~ 40.

Backup and Recovery System:

Introduction:

A computer system likes any mechanical or electrical other may be fail. There are different types of reasons which cause failure are disk failure, system crash, and power failure and so on. In such situations information may be lost. So in advance some action may be taken by database system so that the atomicity and durability properties of transactions are preserved. Database recovery techniques are used for this purpose. The aim of the recovery scheme is to allow database operations to be resumed after a failure with minimum loss of information economically at low cost.

Failure Classification:

The failure of system occurs when the system does not function according to its specifications and fails to deliver the service for which it was intended.

Different types of failure may occur in a system. In simplest type of failure information may be lost and it is very difficult to handle such failure. Various types of failure are**Transaction Failure, System Crash,** and Disk **Failure.**

From the above failures recovery is possible using following techniques

a. **Log Based Recovery**
b. **Shadow Paging**

5.3 Log Based Recovery:

The most widely used structure for recording database modification is the **log**. In this case database modification is maintained through log. Log is a sequence of log records. An update log record describes a single database write and has following fields.

i. **Transaction Identifier**: It is a unique identifier of the transaction that performs write operation.
ii. **Data-item Identifier**: It is a unique identifier of data item written i.e. location on disk of the data item.
iii. **Old Value**: Value of data item before write.
iv. **New Value**: Value of data item after write.

To record significant event during transaction processing some special log records exist these are

- <Ti Start> Transaction Ti has started.
- <Ti, Xj, V1, V2> Transaction Ti has performed a write on data item Xj, Xj had value V1 before the write and will have value V2 after the write.
- <Ti Commit> Transaction Ti has committed.
- <Ti abort> Transaction Ti has aborted.

It is essential that the log record for write be created before the database is modified. If the log record exists we can output the modification to the database. Also we can undo a modification by using old-value field in log record.

5.4 Deferred Update:

In case of system and disk failure log records are useful for recovery for that purpose log record must reside in stable storage. There are two methods to avoid the loss of data using log.

i. Deferred Database Modification

ii. Immediate Database Modification

i. Deferred Database Modification:

In Deferred Database Modification approach, Log is written before any modifications are made to the database. Modifications are not written to the database until after a transaction has reached its commit point.

If the transaction fails before reaching the commit point nothing is required to be done as the database has not been modified. No undoing of changes or rollback will be necessary.

EgwhentransactionTi start then <Ti Start> is written on log. The write(x) operation by the transaction results in writing new record on the log. When transaction partially commits or completes its execution then the record <Ti Commits> is written on log. After this all log records are written on stable storage and then the information is used for executing deferred writes. Following fig.(1) shows the portion of log when the transaction T1 and T2 will be executed.

<T1 start>
<T1, A, 950>
<T1, B, 950>
<T1 Commit>
<T2 start>
<T2, C, 500>
<T2 Commit>

The transaction T1 reads the balance of account A as Rs. 1000 and account B as Rs. 900. Then Rs. 50 withdraws from account A and deposited to account B. Therefore the fig.(1) shows the log record <T1, A, 950> which indicates that the value 950 will be written to data item A at the time of execution of deferred writes. Following fig shows status of log and database at the time of execution of transaction T1 and T2

Log	**Database**
<T1 Start>	
<T1, A, 950>	

<T1, B, 950>
<T1 Commit>

A=950
B=950

<T2 start>
<T2, C, 500>
<T2 Commit>

C=500

In this method when failure occurs then the log is used to handle it, along with recovery procedure Redo(Ti)

Definition of Redo(Ti):

Redo(Ti) sets the value of all data items to its new values updated by transaction Ti. The redo operation must be idempotent i.e. the redo operation should result to the same value when it is executed once or many times.

After a failure occurs the recovery subsystem consults the log to determine which transactions need to be redone. To execute the redo operation the log will be searched for those transactions which are partially committed having the record <Ti Commits> on the log. Those transactions which cannot commit before the failure occurs are not able to execute redo operation.

The redo operation use the record <Ti, Xi, Vj> for setting the value of Vj of the data item Xi updated by transaction Ti.

ii. Immediate Database Modification:

Immediate Database Modification is based on immediate changes to the database after write ( ) operation. It does not wait for commit point.

In this technique the updating of database are done when transaction is in the active state. In case of system crashes or failure the transaction requires old value and new value of data item. In some cases the restores the values of data item to its old value for re-executing the transaction.

When the transaction starts <Ti Start> is written on log. During its execution, any write(x) operation by Ti is preceded by the writing of the appropriate new update to the log. When transaction partially commits then <Ti commits> is written on log. The actual update to the database cannot allow taking place before the corresponding log record is written out to stable storage.

Consider the following fig(1) which shows the log record for T1 and T2 transaction transferring Rs. 50 from account A to account B and reading the balance of account C respectively

<T1 start>
<T1, A, 1000, 950>
<T1, B, 900, 950>
<T1 Commit>
<T2 start>
<T2, C, 500, 500>
<T2 Commit>
Fig(1)

The following fig(2) shows status of log and database transaction T1 and T2

Log	**Database**
<T1 Start>	
<T1, A, 1000, 950>	
<T1, B, 900, 950>	
	A=950
	B=950
<T1 Commit>	
<T2 start>	
<T2, C, 500, 500>	
	C=500
<T2 Commit>	

Fig(2) status of Log and Database

The log is used to handle failure with recovery mechanism of Redo(Ti) and Undo(Ti).

Definition of Redo(Ti)

Redo(Ti) sets value of all data items to its new value updated by transaction Ti.

Definition of Undo(Ti)

Undo(Ti) restores the value of a data item to its old value updated by transaction Ti.

The Undo(Ti) and Redo(Ti) operations are carried out according to following conditions.

1. The transaction Ti needs to be undone if the log contains the record <Ti Start> and does not contains the record <Ti Commit>.
2. The transaction Ti needs to be redone if the log contains the record <Ti Start> and also contains the record <Ti Commit>.

Consider the following execution of transaction T1 and T2. Following Fig(3) shows 3 different conditions of same block after failure occurs.

<T1 Start>	<T1 start>	<T1 start>
<T1, A, 1000, 900>	<T1, A, 1000, 950>	<T1, A, 1000, 950>
<T1, B, 900, 950>	<T1, B, 900, 950>	<T1, B, 900, 950>
	<T1 Commit>	<T1 Commit>
	<T2 start>	<T2 start>
	<T2, C, 500, 500>	<T2, C, 500, 500>
		<T2 Commit>

Fig(3) The same log shown at three different times

So in above fig for the first status of log, if failure occurs then undo(Ti) needs to be executed. For the second case Undo(T2) and Redo(T1) respectively needs to be executed. For third case redo(T1) and redo(T2) need to be executed.

5.5 Shadow paging:

An alternative to log –based recovery technique is shadow paging. The database is portioned into some number of fixed length blocks which are refer to as pages. These pages are stored in any order.*Shadow paging* is an alternative to log-based recovery techniques, which has both advantages and disadvantages. It may require fewer disk accesses, but it is hard to extend paging to allow multiple concurrent transactions. The paging is very similar to paging schemes used by the operating system for memory management. The idea is to maintain two page tables during the life of a transaction: the *current* page table and the *shadow* page table. When the transaction starts, both tables are identical. The shadow page is never changed during the life of the transaction. The current page is updated with each **write**operation. Each table entry points to a page on the disk. When the transaction is committed, the shadow page entry becomes a copy of the current page table entry and the disk block with the old data is released. If the shadow is stored in nonvolatile memory and a system crash occurs, then the shadow page table is copied to the current page table. This guarantees that the shadow page table will point to the database pages corresponding to the state of the database prior to any transaction that was active at the time of the crash, making aborts automatic.

Shadow paging considers

- The database is partitioned into fixed-length blocks referred to as PAGES.
- Page table has n entries – one for each database page.
- Each contain pointer to a page on disk (1 to 1st page on database and so on…).

The idea is to maintain 2 pages tables during the life of transaction.

- The current page table
- The shadow page table

When transaction starts, both page tables are identical

- The shadow page table is never changed over the duration of the transaction.
- The current page table may be changed when a transaction performs a write operation.
- All input and output operations use the current page table to locate database pages on disk.

Advantages

- No Overhead for writing log records.
- No Undo / No Redo algorithm.
- Recovery is faster.

Disadvantages

- Data gets fragmented or scattered.
- After every transaction completion database pages containing old version of modified data need to be garbage collected.
- Hard to extend algorithm to allow transaction to run concurrently.