**CS Fundamental -> DBMS**

# SYLLABUS

**UNIT -1=> Introduction to DBMS and ER data model**

**Data storage:** file processing system, disadvantage; DBMS: need of DBMS: Term Data, database, metaData, data dictionary, database system, database managementSystem, data Abstraction, data independent, system architecture of DBMS dataModel definition ER and relationship data model object oriented object relationModel; ER model, entity set, attributes, primary key, relationship, types of attribute of relationship, role,Cardinality Ratio, Participation constraint, weak EntitySet EER features

**UNIT -2=>Relational data model**

**Relational data model:** terms: relation, schema, attribute, Tuple, domains, relation degree and cardinality, relation in tension and extension, super key, candidate key, primary key and foreign key, relation model constraints, schema diagram, ER to relational mapping, details storage architecture, Magnetic disc RAID Storage access, file & Record organization indexing and orders indices

**UNIT -3=>Relational algebra and relational calculus**

**Selection algebra and its operation:** set theory operation (union, intersection, cartesian product, divisions), relational algebra operates( projection selection join rename

relational calculus TRC DRC

database language SQL ddl dml dcl TCL Features

**UNIT -4=>Normalization**

**Normalization:** Anomalies of un- Normalised relation, need of normalization, pros and cons of normalization, functional dependencies: trivial, full, Partial, transitive multi-value join inclusion dependencies, dependency diagram, inference rule for functional dependencies closer of functionals dependencies, Algorithms to find: 1. candidate key, 2. closer of attributes set, 3. minimal cover of functional dependencies, Normal forms: checking of lossless join decompositions and dependency Preservation, normal forms one and F2 and f3

**UNIT -5=>Transaction management**

**Transaction:** concepts ACID properties, transition state;

schedule: definition, types of schedule, Serializability, Conflicts and view Serializability, Precedence Graph, Recoverable schedule, cascade less schedule, deadlock,

concurrency control protocol: lock based, timestamp base protocol, Recovery system log based recovery checkpoint shadow paging

7. Difference between vertical and horizontal scaling

8. What is sharding

28. Diff between 2 tier and 3 tier architecture

30. Difference between Intension and Extension in a DataBase

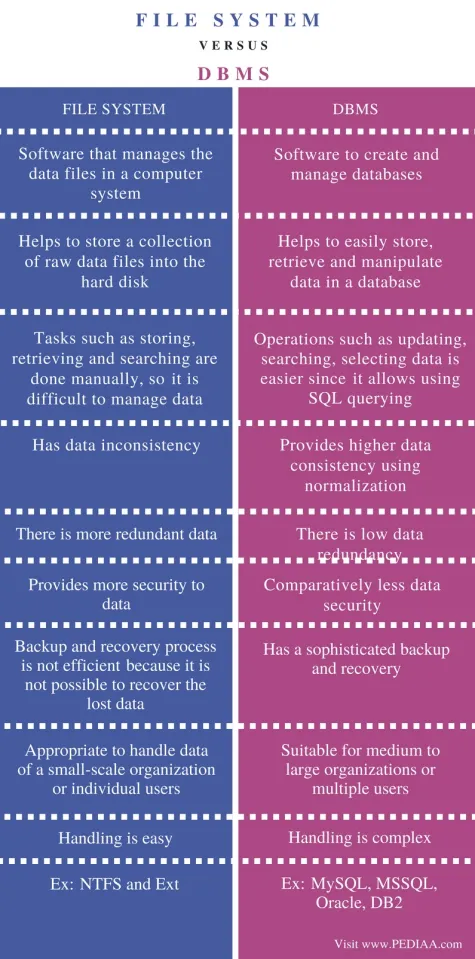
31. Difference between share lock and exclusive lock, definition of lock

# UNIT -1=> Introduction to DBMS and ER data model

## 

## \* File Processing system

File Processing System (FPS) is a way of storing, retrieving and manipulating data which is present in various files. Files are used to store various documents. All files are grouped based on their categories.



**Data =>** Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

**Database=>** A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

## \* **Database Management System (DBMS)**

is a software for storing and retrieving users' data while considering appropriate security measures.

It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

Database management systems were developed to handle the following difficulties of typical File-processing systems supported by conventional operating systems.

1. Data redundancy and inconsistency

2. Difficulty in accessing data

3. Data isolation – multiple files and formats

4. Integrity problems

5. Atomicity of updates

6. Concurrent access by multiple users

7. Security problems

**Data Dictionary**=> A data dictionary contains metadata i.e data about the database. The data dictionary is very important as it contains information such as

what is in the database, who is allowed to access it, where is the database physically stored

## \* Data Abstraction

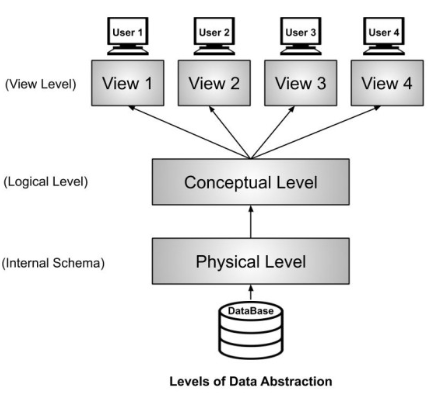
is the ability of the database system to provide abstract view of the data and

* hide certain data details of how data are stored and maintained

**1. Physical level-** PL is also known as Internal schema , It is the lowest level of abstraction that describe how the data are actually stored and shows structure.

**2. Logical level-** LL is also known as the Conceptual level . It is the next high level of abstraction that describes What data are stored in the database and What relationship exist among those data

**3. View level-** It is the highest level of abstraction that describe only part of the entire database



## \* Data Independence

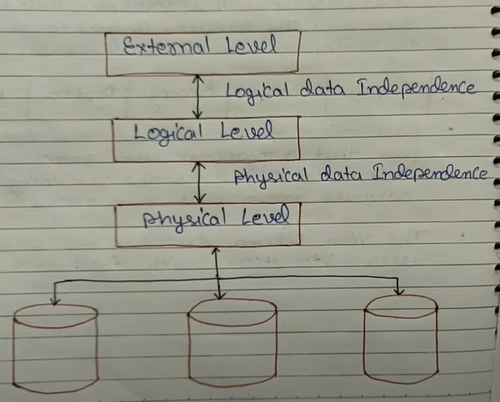
The ability to modify a schema definition in one level without affecting the schema definition in the next higher level is called data independent

* data independent is one of the main advantage of DBMS data independent

Is a two types 1. physical data independent 2.logical data independence

**1. Physical data independence** physical data independence can be defined as the Ability to change the internal schema without having to change the conceptual schema or external schema of Database.

**2. Logical data independence** logical data independent can be defined as the ability to modify the conceptual schema without having to change the external schema or Application program

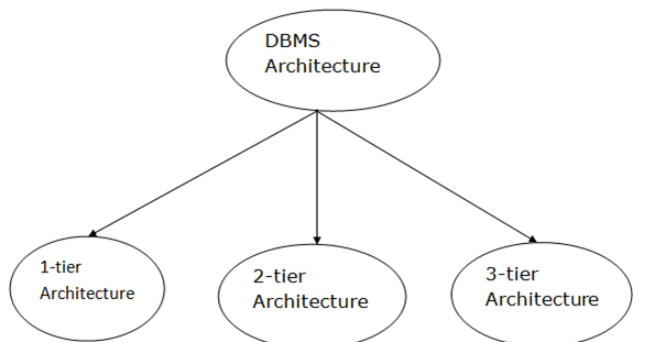


**Database management System Architecture**

DBMS architecture depends upon how users are connected to the database to get their request done.

The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.

## \* Types of DBMS Architecture

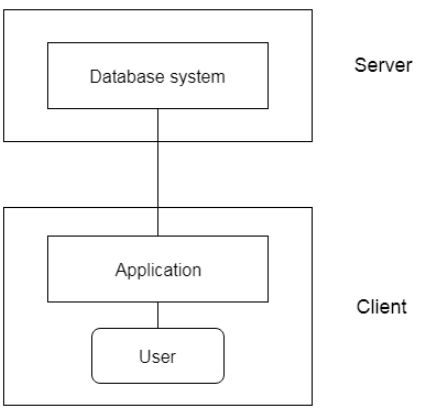


**1) 1-Tier Architecture**

* In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
* Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
* The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

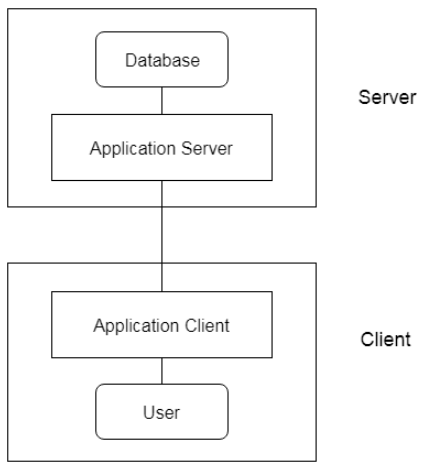
**2) 2-Tier Architecture**

* The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
* The user interfaces and application programs are run on the client-side.
* The server side is responsible to provide the functionalities like: query processing and transaction management.
* To communicate with the DBMS, client-side applications establish a connection with the server side.



**3) 3 - Tier Architecture**

* The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
* The application on the client-end interacts with an application server which further communicates with the database system.
* End user has no idea about the existence of the database beyond the AS application server. The database also has no idea about any other user beyond the application.
* The 3-Tier architecture is used in case of large web applications.



# UNIT -2=>Relational data model

### ER diagram:

●Entity Relationship diagram is a conceptual model that gives the graphical representation of the logical structure of the database.

● It shows all the constraints and relationships that exist among the different components.

● An ER diagram is mainly composed of following three components-

Entity Sets, Attributes and Relationship Set.

**Entity Set:**

An entity set is a set of the same type of entities.



### Relationship:

A relationship is defined as an AASE **association among several entities**.

● **Unary Relationship Set** - Unary relationship set is a relationship set where only one entity set participates in a relationship set.

● **Binary Relationship Set** - Binary relationship set is a relationship set where two entity sets participate in a relationship set.

● **Ternary Relationship Set** - Ternary relationship set is a relationship set where three entity sets participate in a relationship set.

● **N-ary Relationship Set** - N-ary relationship set is a relationship set where ‘n’ entity sets participate in a relationship set.

### Cardinality Constraint:

Cardinality constraint defines the maximum number of relationship instances in which an entity can participate.

● **One-to-One Cardinality** - An entity in set A can be associated with at most one entity in set B.

An entity in set B can be associated with at most one entity in set A.

● **One-to-Many Cardinality** - An entity in set A can be associated with any number (zero or more) of entities in set B. An entity in set B can be associated with at most one entity in set A.

● **Many-to-One Cardinality** - An entity in set A can be associated with at most one entity in set B. An entity in set B can be associated with any number of entities in set A.

● **Many-to-Many Cardinality** - An entity in set A can be associated with any number (zero or more) of entities in set B. An entity in set B can be associated with any number (zero or more) of entities in set A.

**Generalization** is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.

In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

**Specialization** is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.

Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

### **Attributes:** Attributes are the descriptive properties which are owned by each entity of an Entity Set.

**Types of Attributes: (5)**

● **Simple Attributes** - Simple attributes are those attributes which cannot be divided further. Ex. Age

● **Composite Attributes** - Composite attributes are those attributes which are composed of many other simple attributes. Ex. Name, Address

● **Multi Valued Attributes** - Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set. Ex. Mobile No, Email ID

● **Derived Attributes** - Derived attributes are those attributes which can be derived from other attribute(s). Ex. Age can be derived from DOB.

● **Key Attributes** - Key attributes are those attributes which can identify an entity uniquely in an entity set. Ex. Roll No.

### **Constraints:** Relational constraints are the restrictions imposed on the database contents and operations. They ensure the correctness of data in the database.

● **Domain Constraint** - Domain constraint defines the domain or set of values for an attribute. It specifies that the value taken by the attribute must be the atomic value from its domain.

● **Tuple Uniqueness Constraint** - Tuple Uniqueness constraint specifies that all the tuples must be necessarily unique in any relation.

● **Key Constraint** - All the values of the primary key must be unique. The value of the primary key must not be null.

● **Entity Integrity Constraint** - Entity integrity constraint specifies that no attribute of primary key must contain a null value in any relation.

● **Referential Integrity Constraint** - It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null.

# UNIT -3=>Relational algebra and relational calculus

### Intro

Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.

**Domain:** It contains a set of atomic values that an attribute can take.

**Attribute:** It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai)

**Relational instance:** In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.

**Relational schema:** A relational schema contains the name of the relation and name of all columns or attributes.

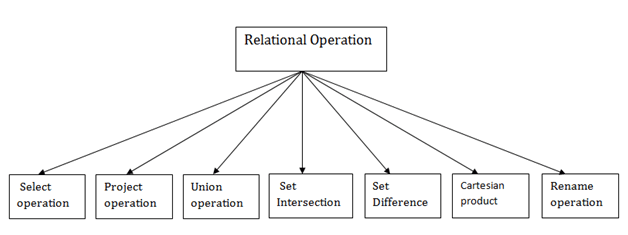
**Relational key:** In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

## 

### 3.1 Algebra and its operation

Relational algebra is a (PQL) procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries. Adv- query optimizer

[Types of Relational operation](https://www.javatpoint.com/dbms-relational-algebra) (IF YOU WANT TO SEE EXAMPLE THEN USE THIS LINK )



**1. Selection** => It is denoted by sigma (σ). The select operation selects tuples that satisfy a given predicate.

**2. Projection** => It is denoted by ∏. This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.

**3. Union** => It is denoted by ∪. It eliminates the duplicate tuples. Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.

**4. Set Intersection** => It is denoted by intersection ∩. Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.

**5. Set Difference =>** It is denoted by intersection minus (-). Suppose there are two tuples R and S. The set intersectionoperation contains all tuples that are in R but not in S.

**6. Cartesian Product**=> It is denoted by X. It is also known as a cross product. The Cartesian product is used to combine each row in one table with each row in the other table.

**7. Rename Operation**=> It is denoted by **rho** (ρ). The rename operation is used to rename the output relation.

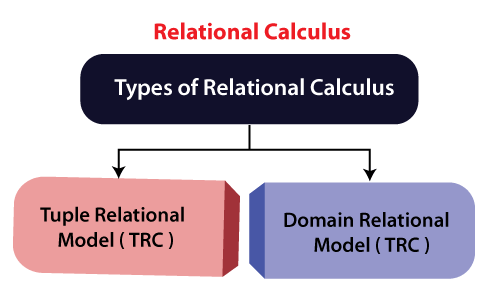
### 3.2 Relation Calculus [Exp](https://www.youtube.com/watch?v=2HBwczTIKI8&ab_channel=Education4u)

It is an alternative way of Relation Algebra, Relational calculus is a non-procedural query language.

In the non-procedural query language, the user is concerned with the details of how to obtain the end results.

* The relational calculus tells what to do but never explains how to do.

**Types of Relational Calculus**



**1) Tuple Relational Calculus**

=> It is a non-procedural query language which is based on finding a number of tuple variables also known as range variables for which the predicate holds true.

-It describes the desired information without giving a specific procedure for obtaining that information.

- The tuple relational calculus is specified to select the tuples in a relation.

- In TRC, the filtering variable uses the tuples of a relation. The result of the relation can have one or more tuples.

**2) Domain Relational Calculus**

=>The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes.

-Domain relational calculus uses the same operators as tuple calculus.

-It uses logical connectives ∧ (and), ∨ (or) and ┓ (not). It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable.

-The QBE or Query by example is a query language related to domain relational calculus.

**NEXT TOPIC IS => SQL JOIN - SQL COMMANDS That I already covered in SQL so refer from there.**

# UNIT -4=>Normalization

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
* Normalization divides the larger table into smaller and links them using relationships.
* The normal form is used to reduce redundancy from the database table.

**-Why do we need Normalization?**

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

### Anomalies can be categorized into three types:

* **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
* **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
* **Updation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

### Types of Normal Forms

**1) First Normal form 1NF**=> A relation will be 1NF if it contains an atomic value.

* It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
* First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

**2) Second Normal form 2NF**=>In the 2NF, relational form must be in 1NF.

* In the second normal form, all non-key attributes are fully functional dependent on the primary key

**3) Third Normal form 3NF**=> A relation will be in 3NF if it is in 2NF and does not contain any transitive partial dependency.

* 3NF is used to reduce data duplication. It is also used to achieve data integrity.
* If there is no transitive dependency for non-prime attributes, then the relation must be in the third normal form.

**4) BCNF**=> BCNF is the advanced 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.

**5) Forth Normal form 4NF**=>A relation will be in 4NF if it is in Boyce Codd normal form and has no multivalued dependency.

* For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

**Advantages of Normalization**

* Normalization helps to minimize data redundancy.
* Greater overall database organization.
* Data consistency within the database.
* Much more flexible database design.
* Enforces the concept of relational integrity.

**Disadvantages of Normalization**

* You cannot start building the database before knowing what the user needs.
* The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
* It is very time-consuming and difficult to normalize relations of a higher degree.
* Careless decomposition may lead to a bad database design, leading to serious problems.

**Closure of an Attribute Set:**

The set of all those attributes which can be functionally determined from an attribute set is called a closure of that attribute set.

### **Keys:** A key is a set of attributes that can identify each tuple uniquely in the given relation.

**Types of Keys:**

● **Super Key** - A superkey is a set of attributes that can identify each tuple uniquely in the given relation. A super key may consist of any number of attributes.

● **Candidate Key** - A set of minimal attribute(s) that can identify each tuple uniquely in the given relation is called a candidate key.

● **Primary Key** - A primary key is a candidate key that the database designer selects while designing the database. Primary Keys are unique and NOT NULL.

● **Alternate Key** - Candidate keys that are left unimplemented or unused after implementing the primary key are called as alternate keys.

● **Foreign Key** - An attribute ‘X’ is called as a foreign key to some other attribute ‘Y’ when its values are dependent on the values of attribute ‘Y’. The relation in which attribute ‘Y’ is present is called as the referenced relation. The relation in which attribute ‘X’ is present is called as the referencing relation.

● **Composite Key** - A primary key composed of multiple attributes and not just a single attribute is called a composite key.

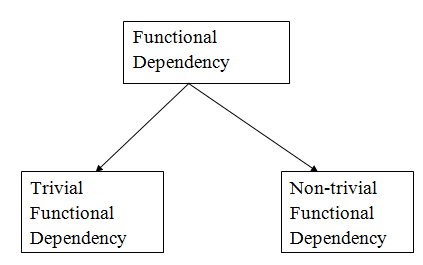
● **Unique Key** - It is unique for all the records of the table. Once assigned, its value cannot be changed i.e. it is non-updatable. It may have a NULL value.

### Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

X → Y

Types of Functional dependency



**1. Trivial functional dependency**

* A → B has trivial functional dependency if B is a subset of A.
* The following dependencies are also trivial like: A → A, B → B

**2. Non-trivial functional dependency**

* A → B has a non-trivial functional dependency if B is not a subset of A.
* When A intersection B is NULL, then A → B is called as complete non-trivial.

**Multivalued Dependency**

* Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
* A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

### Inference Rule(IR)

* Armstrong's axioms are the basic inference rule.
* Armstrong's axioms are used to conclude functional dependencies on a relational database.
* The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
* Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

**1) Reflexive Rule(IR1)**=> In the reflexive rule, if Y is a subset of X, then X determines Y.

Ex=If X ⊇ Y then X → Y

**2)Augmentation Rule(IR2**)=>The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

Ex= If X → Y then XZ → YZ

**3) Transitive Rule(IR3**)=>In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

Ex= If X → Y and Y → Z then X → Z

**4)Union Rule(IR4)**=>Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

Ex= If X → Y and X → Z then X → YZ

**5) Decomposition Rule(IR5)** =>Decomposition rule is also known as project rule. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

Ex= If X → YZ then X → Y and X → Z

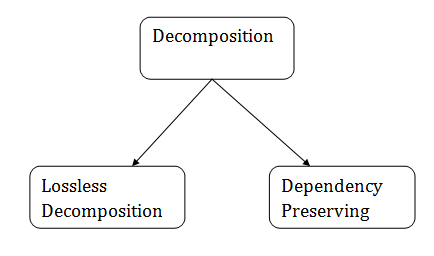
**6) Pseudo Transitive Rule(IR6)**=>In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

Ex= If X → Y and YZ → W then XZ → W

### Relational Decomposition

* When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
* In a database, it breaks the table into multiple tables.
* If the relation has no proper decomposition, then it may lead to problems like loss of information.
* Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

Types of Decomposition



**Lossless Decomposition**

* If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
* The lossless decomposition guarantees(LDG) that the join of relations will result in the same relation as it was decomposed.
* The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

**Dependency Preserving**

* It is an important constraint of the database.
* In the dependency preservation, at least one decomposed table must satisfy every dependency.
* If a relation R is decomposed into relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or must be derivable from the combination of functional dependencies of R1 and R2.
* For example, suppose there is a relation R (A, B, C, D) with functional dependency set (A->BC). The relational R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of relation R1(ABC).

# 

# 

# UNIT -5=>Transaction management

### ACID Properties

To ensure the consistency of the database, certain properties are followed by all the transactions occurring in the system.

These properties are called as ACID Properties of a transaction.

**1) Atomicity** – o This property ensures that either the transaction occurs completely or it does not occur at all.

o In other words, it ensures that no transaction occurs partially.

**2) Consistency** – o This property ensures that integrity constraints are maintained.

o In other words, it ensures that the database remains consistent before and after the transaction.

**3) Isolation** – o This property ensures that multiple transactions can occur simultaneously without causing any inconsistency.

o The resultant state of the system after executing all the transactions is the same as the state that would be achieved if the transactions were executed serially one after the other.

**4) Durability** – o This property ensures that all the changes made by a transaction after its successful execution are written successfully to the disk.

o It also ensures that these changes exist permanently and are never lost even if there occurs a failure of any kind.

### Schedules:

The order in which the operations of multiple transactions appear for execution is called as a schedule.

**● Serial Schedules** – o All the transactions execute serially one after the other.

o When one transaction is executing , no other transaction is allowed to execute.

o Serial schedules are always- Consistent, Recoverable, Cascadeless and Strict.

**● Non-Serial Schedules** – o Multiple transactions execute concurrently.

o Operations of all the transactions are interleaved or mixed with each other.

o Non-serial schedules are not always- Consistent, Recoverable, Cascadeless and Strict.

**Serializability** – ● Some non-serial schedules may lead to inconsistency of the database.

● Serializability is a concept that helps to identify which non-serial schedules are correct and will maintain the consistency of the database.

**● Serializable Schedules** – o If a given non-serial schedule of ‘n’ transactions is equivalent to some serial schedule of ‘n’ transactions, then it is called as a serializable schedule.

o Serializable schedules are always- Consistent, Recoverable, Cascadeless and Strict.

### Types of Serializability –

1● **Conflict Serializability** - If a given NSS non-serial schedule can be converted into a SS serial schedule by swapping its non-conflicting operations, then it is called a conflict serializable schedule.

2● **View Serializability** - If a given schedule is found to be viewed as equivalent to some serial schedule, then it is called a view serializable schedule.

3. **Non-Serializable Schedules** –

● A non-serial schedule which is not serializable is called a non-serializable schedule.

● A non-serializable schedule is not guaranteed to produce the same effect as produced by some serial schedule on any consistent database.

● Non-serializable schedules- may or may not be consistent, may or may not be recoverable.

4● **Irrecoverable Schedules** – If in a schedule, o A transaction performs a dirty read operation from an uncommitted transaction o And commits before the transaction from which it has read the value then such a schedule is known as an Irrecoverable Schedule.

5● **Recoverable Schedules** – If in a schedule, o A transaction performs a dirty read operation from an uncommitted transaction o And its commit operation is delayed till the uncommitted transaction either commits or roll backs then such a schedule is known as a Recoverable Schedule.

### Types of Recoverable Schedules –

● **Cascading Schedule** - If in a schedule, failure of one transaction causes several other dependent transactions to rollback or abort, then such a schedule is called as a Cascading Schedule or Cascading Rollback or Cascading Abort.

● **Cascadeless Schedule** - If in a schedule, a transaction is not allowed to read a data item until the last transaction that has written it is committed or aborted, then such a schedule is called a Cascadeless Schedule.

● **Strict Schedule** - If in a schedule, a transaction is neither allowed to read nor write a data item until the last transaction that has written it is committed or aborted, then such a schedule is called a Strict Schedule.

### Concurrency Control Protocol

It is the process of managing simultaneous execution of transaction in shared database to ensure serializability of transaction.

Purpose of concurrency control

-to enforce isolation

-to preserve database consistency

-to resolve read write and write write conflicts

**Concurrency control technique**

**1. Lock based protocol:** A lock guarantees exclusive use of a data item to a current transaction

-to access data item.(Lock Acquire)

-After completion of transaction (release lock)

\* –-Types of locks

**1. Shared lock:**

* It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction.
* It can be shared between the transactions because when the transaction holds a lock, then it can't update the data on the data item.

**2. Exclusive lock:**

* In the exclusive lock, the data item can be both reads as well as written by the transaction.
* This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

**2. Timestamp ordering protocol**

* The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.
* The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.
* The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.
* Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at 009 times. T1 has the higher priority, so it executes first as it is entered the system first.
* The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

**3. Log Based Recovery**

* The log is a sequence of records. Log of each transaction is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
* If any operation is performed on the database, then it will be recorded in the log.
* But the process of storing the logs should be done before the actual transaction is applied in the database.

**4. recovery checkpoint Shadow Paging**

**Shadow Paging** is recovery technique that is used to recover [database](https://www.geeksforgeeks.org/what-is-database/).

Maintain two page table during the life cycle of transaction

when transaction starts both page table are identical

shadow page is never changed over duration of transaction

current page table may change during write operation

all input and output operations use the current table to locate database page on disc store

shadow page table in non volatile storage when transaction commits system right current page table to non volatile storage the current page table then becomes new shadow page table

**Advantage-**

In this method, recovery from crash is inexpensive and quite fast.

There is no need of operations like- Undo and Redo.

**Disadvantage**

* The commit of single transaction requires multiple blocks which decreases execution speed.
* To allow this technique to multiple transactions concurrently it is difficult.

### Indexing in DBMS

* Indexing is used to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed.
* The index is a type of data structure. It is used to locate and access the data in a database table quickly.