UNIT-I

I N T R O D U C T I O N

**1. What is Database (DB)?**

**Ans:**

A **Database** is a logically coherent collection of data with some inherent meaning, representing some aspect of the real world and which is designed, built and populated with **data** for a ***specific purpose.***

**2. What is a Database Management System?**

**Ans:**

A **Database Management System** (**DBMS**) is a set of ***computer programs*** that controls the creation, maintenance, and the use of the database of an organization and its end users. It allows organizations to place control of organization-wide database development in the hands of database administrators (DBAs) and other specialists.

##### “A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.”

**3. What is Database System?**

**Ans:**

The ***database*** and ***DBMS software*** together is called the Database System.

**4 What is Database Administrator (DBA)?**

**Ans:**

As system controls increase usability of the system decreases. Actually it is perfectly possible to have an efficient and reliable system which no one can use effectively. This is never the explicit goal of the DBA, but there is a danger that it is an implicit goal.

The person who looks after the database needs to balance all needs of the users, whether they know they need it or not. No user wants security, for instance, yet if someone hacks in and deletes all their work the DBA becomes the target. Perfectly designed security is completely invisible to the valid user, but is automatic and total for the invalid user.

Security is not the only issue of importance for the DBA. They are also concerned with:

* System performance and tuning
* Data backup and recovery
* Product and tool selection, installation, and maintenance
* System documentation
* Support
* Education
* Fortune Telling / Future Prediction

A good DBA is almost never seen. The fact is that if you have to phone the DBA then the DBA has failed. The system will be monitored continuously, and problems detected and fixed before they are noticed by users. Long term issues, such as data growth, diversification, the addition of new projects, do need to be discussed with the DBA, but the DBA should be able to detect most issues anyway and handle them transparently from the users and developers.

“One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a database administrator (DBA). The functions of a DBA include:

• **Schema definition**: The DBA creates the original database schema by executing a set of data definition statements in the DDL.

• **Storage structure and access-method definition**.

**• Schema and physical-organization modification**: The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

• **Granting of authorization for data access**: By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.

• **Routine maintenance**: Examples of the database administrator’s routine

maintenance activities are:

􀀀 Periodically backing up the database, either onto tapes or onto remote

servers, to prevent loss of data in case of disasters such as flooding.

􀀀 Ensuring that enough free disk space is available for normal operations,

and upgrading disk space as required.

􀀀 Monitoring jobs running on the database and ensuring that performance

is not degraded by very expensive tasks submitted by some users.”

**5. Where the DBMS is used?**

**Ans:** Databases are widely used in,

1. Banking
2. Airlines
3. Universities
4. Credit card transactions
5. Telecommunication
6. Finance
7. Sales
8. On-line retailers
9. Manufacturing
10. Human resources

**6. What is the purpose of DBMS?**

**Ans:**

1. To see why database management systems are necessary, let us look at a typical **“file-processing system”** supported by a conventional operating system.

The application is a savings bank:

* Savings account and customer records are kept in permanent system files.
* Application programs are written to manipulate files to perform the following tasks:
* Debit or credit an account.
* Add a new account.
* Find an account balance.
* Generate monthly statements.

1. **Development of the system proceeds** as follows:

* New application programs must be written as the need arises.
* New permanent files are created as required.
* But over a long period of time files may be in different formats, and
* Application programs may be in different languages.

1. So we can see there are **problems with the straight file-processing approach**:

* **Data redundancy and inconsistency**
* Same information may be duplicated in several places.
* All copies may not be updated properly.
* **Difficulty in accessing data**
* May have to write a new application program to satisfy an unusual request.
* e.g. find all customers with the same postal code.
* Could generate this data manually, but a long job.
* **Data isolation**
* Data in different files.
* Data in different formats.
* Difficult to write new application programs.
* **Multiple users**
* Want concurrency for faster response time.
* Need protection for concurrent updates.
* e.g. two customers withdrawing funds from the same account at the same time - account has $500 in it, and they withdraw $100 and $50. The result could be $350, $400 or $450 if no protection.
* **Security problems**
* Every user of the system should be able to access only the data they are permitted to see.
* e.g. payroll people only handle employee records, and cannot see customer accounts; tellers only access account data and cannot see payroll data.
* Difficult to enforce this with application programs.
* **Integrity problems**
* Data may be required to satisfy constraints.
* e.g. no account balance below $25.00.
* Again, difficult to enforce or to change constraints with the file-processing approach.

These problems and others led to the development of ***database management system.***

**7. What are the Advantages of DBMS?**

**Ans:**

1. Redundancy is controlled.
2. Unauthorised access is restricted.
3. Providing mutiple user interfaces.
4. Enforcing integrity constraints.
5. Providing backup and recovery.
6. Provide security for the database.

**8. What are the Disadvantages of File Processing System?**

**Ans:**

1. Data redundancy & inconsistency.
2. Difficult to accessing data.
3. Data isolation.
4. Data integrity.
5. Concurrent access is not possible.
6. Security problems

**9. Describe the three levels of data abstraction?**

**Ans:**

There are ***three levels*** of abstractions:

**Physical level🡪** The lowest level of abstraction ***describes how data are stored.***

* How the data are stored.
* E.g. index, B-tree, hashing.
* Lowest level of abstraction.
* Complex low-level structures described in detail.

**Logical level or Conceptual level🡪** The next higher level of abstraction, ***describes what data are stored in database and what relationship among those data.***

* Next highest level of abstraction.
* Describes what data are stored.
* Describes the relationships among data.
* Database administrator level.

**View level🡪** The highest level of abstraction ***describes only part of entire database.***

* Highest level.
* Describes part of the database for a particular group of users.
* Can be many different views of a database.
* e.g. tellers in a bank get a view of customer accounts, but not of payroll data.

View n

View 2

View 1

……..

# Logical level

Physical level

**10. Define the “Integrity Rules”.**

**Ans:**

There are ***two*** Integrity rules.

**Entity Integrity:** States that “Primary Key **cannot** have **NULL** value”.

**Referential Integrity:** States that “Foreign Key **can** be either a **NULL** value or should be **Primary Key value** of other relation”.

11. What is Data Independence?

**Ans:**

Data Independence means that ***“The*** ***application is independent of the storage structure and access strategy of data”.*** In other words, the ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

***Physical Data Independence:*** Modification in physical level should not affect the logical level.

* The ability to modify the physical scheme without causing application programs to be rewritten.
* Modifications at this level are usually to improve performance.

***Logical Data Independence:***Modification in logical level should affect the view level.

* The ability to modify the conceptual scheme without causing application programs to be rewritten.
* Usually done when logical structure of database is altered.

NOTE: Logical Data Independence is more difficult to achieve.

12. What is a view? How it is related to data independence?

Ans:

A view may be thought of as a *virtual table*, that is, a table that *does not really exist* in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

13. What is Data Model?

Ans:

A collection of conceptual tools for *describing data, data relationships, data semantics and constraint is the data model.*

14. Short Note :Relational Model.

Ans:

A database based on the relational model developed by *E. F. Codd*. A relational database allows the definition of *data structures, storage and retrieval operations and integrity constraints*. In such a database, the data and relations between them are *organized* in tables. A table is a collection of records and each record in a table contains the same fields.

Properties of Relational Tables:

* Values are Atomic.
* Each Row is Unique.
* Column values are of the same kind
* The Sequence of Columns is Insignificant.
* The Sequence of Rows is Insignificant.
* Each Column has a Unique Name.

Certain fields may be designated as keys, which means that searches for specific values of that field will use indexing to speed them up. Where fields in two different tables take values from the same set, a join operation can be performed to select related records in the two tables by matching values in those fields. Often, but not always, the fields will have the same name in both tables. For example, an "orders" table might contain (customer-ID, product-code) pairs and a "products" table might contain (product-code, price) pairs so to calculate a given customer's bill you would sum the prices of all products ordered by that customer by joining on the product-code fields of the two tables. This can be extended to joining multiple tables on multiple fields. Because these relationships are only specified at retrieval time, relational databases are classed as dynamic database management system. The relational database model is based on the Relational Algebra.

15. Short Note : Object/Relational Model.

**Ans:**

Object/Relational Database Management Systems (ORDBMSs) ***add new object storage capabilities*** to the relational systems at the core of modern information systems. These new facilities integrate management of traditional fielded data, complex objects such as time-series and geospatial data and diverse binary media such as audio, video, images, and applets. By encapsulating methods with data structures, an ORDBMS server ***can execute complex analytical and data manipulation operations*** to search and transform multimedia and other complex objects.

As an evolutionary technology, the object/relational (OR) approach has inherited the robust transaction- and performance-management features of it is relational ancestor and the flexibility of its object-oriented cousin. Database designers can work with familiar tabular structures and data definition languages (DDLs) while assimilating new object-management possibilities. Query and procedural languages and call interfaces in ORDBMSs are familiar: SQL3, vendor procedural languages, and ODBC, JDBC, and proprietary call interfaces are all extensions of RDBMS languages and interfaces. And the leading vendors are, of course, quite well known: IBM, Inform ix, and Oracle.

16. What is Structured Query Language?

Ans:

SQL is a standard language for accessing and manipulating databases.

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* SQL is an ANSI (American National Standards Institute) standard

## 17. What can SQL do?

## Ans:

* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database
* SQL can create stored procedures in a database
* SQL can create views in a database
* SQL can set permissions on tables, procedures, and views

18. What is Entity Relationship Model (E-R Model)?

Ans:

This data model is based on real world that consists of basic objects called *Entities* and of relationship among these objects. Entities are described in a database by a *set of attributes.*

19. What is Entity Relationship Diagram (E-R Diagram)?

Ans:

In ER modeling, the structure for a database is portrayed as a diagram, called an entity-relationship diagram (or ER diagram), that resembles the graphical breakdown of a sentence into its grammatical parts. Entities are rendered as points, polygons, circles, or ovals. Relationships are portrayed as lines connecting the points, polygons, circles, or ovals. Any ER diagram has an equivalent relational table, and any relational table has an equivalent ER diagram. ER diagramming is an invaluable aid to engineers in the design, optimization, and debugging of database programs.

20. What is Object Oriented Model?

Ans:

This model is based on *collection of objects.* An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object. These bodies of code are called *methods.* Objects that contains same types of values and the same methods are grouped together into *classes*.

21. What are the Phases of System Database Design?

Ans:

**i) Conceptual design 🡪**produces conceptual ER diagram. It's objective is to represent the real world using the ER model. In this design phase, the designer (modeler) should focus on capturing the application requirements and representing the applications in a natural and straight forward way.

**ii) Logical design** 🡪 attempts to map a conceptual ER model to a format that can be acceptable by a database system. This step deals with implementation. You need first to determine which database system is to be used. This issue is simplified because relational databases are the default choice. The next step is to map the conceptual ER model to relations (tables) so that you can improve the relations in terms of normalization and enter them into a specific database system.

22. What is an Entity?

Ans:

It is a ‘thing’ in the real world with an independent existence.

23. What is an Entity Type?

Ans:

It is a *collection (set) of entities* that have same attributes.

24. What is an Entity Set?

Ans:

It is a *collection of all entities* of particular entity type in the database.

25. What is an Extension of Entity Type?

Ans:

The collections of entities of a particular entity type are *grouped* together into an entity set.

26 . What is Weak Entity Set?

Ans: An entity set may *not have sufficient attributes to form a primary key*, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be *Weak Entity Set.*

* The existence of a weak entity set depends on the existence of a strong entity set; it must relate to the strong set via a on-to-many relationship set.
* A discriminator of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
* The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence depend, plus the weak entity set’s discriminator.

27. What is Strong Entity Set?

Ans:

One that does *have a primary key* is called a strong entity set.

28. What is an Attribute?

Ans:

It is a particular property, which *describes the entity.*

29. What is Domain of Attributes?

Ans:

The *set of possible values* for an attribute is called the domain of the attribute.

Example–   
i) The domain of attribute marital status is having four values: single,

married, divorced or widowed.

ii) The domain of the attribute month is having twelve values ranging from

January to December.

30. Describe different types of Attributes?

Ans:

1. Key Attribute🡪 The attribute (or combination of attributes) that is *unique* for every entity instance.

Example – The account number of an account, the employee id of an employee etc.

1. Simple Attribute🡪 The attribute that *cannot be divided* into simpler components.

Example – Age of an employee

1. Composite Attribute🡪 The attribute that *can be split* into components.

Example – Date of joining of the employee. Can be split into day, month and year.

1. Single Valued Attribute🡪 The attribute that can take on *only a single value* for each entity instance.

Example – Age of employee. There can be only one value for this.

1. Multi-Valued Attribute🡪 The attribute that can *take up many values.*

Example – Skill set of employee.

1. Stored Attribute🡪 The attribute that need to be *stored permanently.*

Example – Name of an employee.

vii) Derived Attribute🡪 The attribute that *can be calculated* based on

other attributes.

Example – Years of service of employee can be calculated from date of joining and current date.

31. What is Relationship?

Ans:

A relationship is an *association* between several entities.

32. What is Relationship Set?

Ans: A relationship set is a set of relationships of the same type.

33. What is a Relation Schema and a Relation?

Ans:

A relation Schema denoted by R (A1, A2, …, An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, …, tn). Each tuple is an ordered list of n-values

t= (v1, v2, …, vn).

34. What is degree of a Relation?

Ans:

It is the *number of attribute* of its relation schema.

35. What is Relationship Type?

Ans:

Relationship type defines a *set of associations* or a relationship set among a given set of entity types.

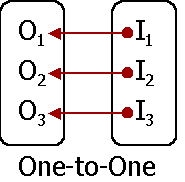
36. What is degree of Relationship Type?

It is the *number of entity type* participating.

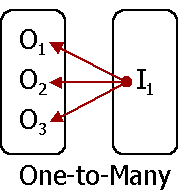
37. What is the Mapping Constraints?

**Ans:** An E-R Diagram scheme may define certain constraints to which the contents of a database must conform.

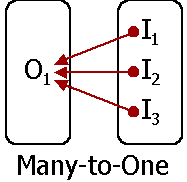
* **Mapping Cardinalities:** express the number of entities to which another entity can be associated via a relationship. For binary relationship sets between entity sets A and B, the mapping cardinality must be one of:
  1. **One-to-one**: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



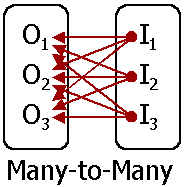
* 1. **One-to-many**: An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.



* 1. **Many-to-one**: An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.



4.**Many-to-many**: Entities in A and B are associated with any number from each other.



The appropriate mapping cardinality for a particular relationship set depends on the real world being modeled.

* **Existence Dependencies:** If the existence of entity X depends on the existence of entity Y, then X is said to be **existence dependent** on Y.
* Y is called dominant entity
* X is called subordinate entity

**38. What is Key / Index?**

**Ans:**

An ***index or key*** is an attribute or collection of attributes that may be used to identify or retrieve one or more records.

39. What is Super key?

Ans:

A super key is a set of one or more attributes which; taken collectively, allow us to identify uniquely an entity in the entity set.

40. What is Candidate Key?

Ans:

A superkey may contain extraneous attributes, and we are often interested in the smallest superkey. A superkey for which no subset is a superkey is called a candidate key.

41. What is Primary Key?

Ans:

A primary key is a candidate key (there may be more than one) chosen by the DB designer to identify entities in an entity set.

**Just Remember:** The primary key of a weak entity is found by taking the primary key of the strong entity on which it is existence-dependent, plus the discriminator of the weak entity set.

42. Describe the major components of Entity-Relationship Diagram?

Ans:

* **Rectangles**, which represent entity sets
* **Ellipses**, which represent attributes
* **Diamonds**, which represent relationship sets
* **Lines**, which link attributes to entity sets and entity sets to relationship sets
* **Double ellipses**, which represent multi-valued attributes
* **Dashed ellipses**, which denote derived attributes
* **Double lines**, which indicate total participation of an entity in a relationship set
* **Double rectangles**, which represent weak entity sets

The symbols of the components of Entity-Relationship Diagrams are,

E

Entity set Attribute

E

Weak entity set Multi-Valued Attribute

R

Relationship set Derived Attribute

R

R

### Identifying Relationship Total

E

R

### set for Weak Entity participation of

### entity set in

### relationship

Discriminating

### Primary key attribute of weak entity

### set

R

R

many-to-many many-to-one

relationship relationship

R

one-to-one 1..h cardinality

E

R

relationship limits

role-

ISA

R

name role ISA

E

## indicator (Generalization or

## Specialization)

## Total generalization disjoint

ISA

ISA

## generalization

## disjoint

## 

## 

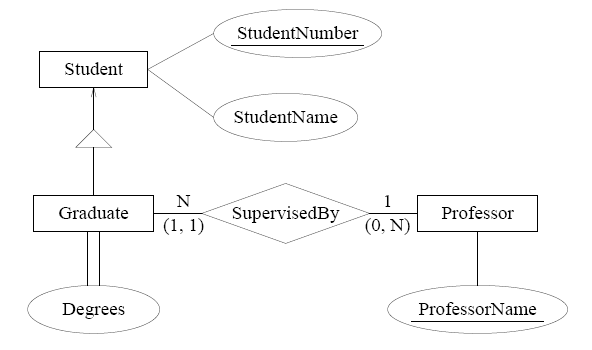
## 43. What are Extended E-R Features?

## Ans:

## These are *four* types:

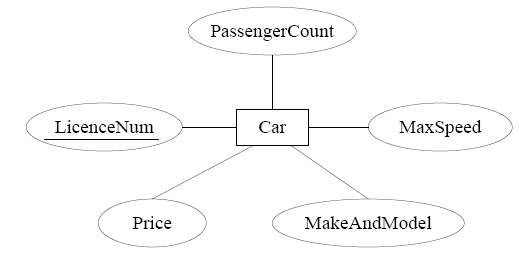
## i) Specialization🡪 When there are distinct subgroups within an entity set, the subgroups can be diagrammed with the bottom half of a diamond. When referencing the subgroups or lower-level entity sets, the specialization is used.

**Example --** Graduate students are students that have a supervisor and a number of degrees.



**ii) Generalization🡪** Two or more existing entity sets can be abstracted as a more general kind of entity set.

**Example --** A vehicle abstracts the notion of a car and a truck.



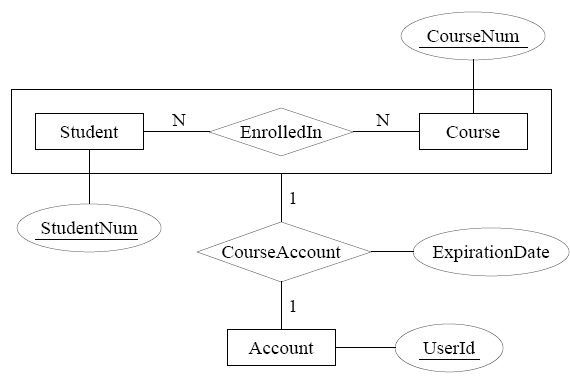
## 

## 

## 

**iii)** **Aggregation🡪** *Aggregation* is an important concept in database design where composite objects can be modeled during the design of database applications. Therefore, preserving the aggregation concept in database implementation is essential. In this paper, we propose models for implementation of aggregation in an *Object-Relational Database Management System* (ORDBMS) through the use of *index clusters* and *nested tables*. ORDBMS is a commercial Relational Database Management Systems (RDBMS), like Oracle, which support some object-oriented concepts. We will also show how queries can be performed on index clusters and nested tables.

**Example --** Accounts are assigned to a given student enrollment.



**iv) Structured Attributes🡪**

1. **Composite Attribute**🡪Can be split into components.

**Example --** Date of joining of the employee. Can be split into day, month and year.

1. **Multi-valued Attribute**🡪 Can take up many values.

**Example -- S**kill set of employee

**44. What is Data Dictionary?**

**Ans:**

A data dictionary is a ***collection of descriptions of the data objects*** or items in a data model for the benefit of programmers and others who need to refer to them. A first step in analyzing a system of objects with which users interact is to identify each object and its relationship to other objects. This process is called ***data modeling*** and results in a picture of object relationships. After each data object or item is given a descriptive name, its relationship is described (or it becomes part of some structure that implicitly describes relationship), the type of data (such as text or image or binary value) is described, possible predefined values are listed, and a brief textual description is provided. This collection can be organized for reference into a book called a data dictionary.

A data dictionary is organized into five sections:

* Data elements
* Data flows
* Data Stores
* Processes
* External Entities

When developing programs that use the data model, a data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain, and basically what the data item means in real-world terms. For example, a bank or group of banks could model the data objects involved in consumer banking. They could then provide a data dictionary for a bank's programmers. The data dictionary would describe each of the data items in its data model for consumer banking (for example, “Account holder” and “Available credit”).

**RELATIONAL MODEL**

52. Describe the Structure of Relational Databases?

Ans:

The data in the database is represented as tables and is considered as structural aspect of the model. In order to study the structure of the relational database the following concepts are essential:

* Structural Specification: The data in the database is viewed as tables
* Integrity Specification: Tables follow a mechanism called Integrity Constraints to prevent in valid data entry into the base tables of the database.
* Manipulative Specification: Operators manipulates tables for the purposes of data retrieval. New tables can be derived from the given tables.

53. What is Relation?

Ans:

A relation is a two dimensional table where columns are representing attributes of the relation and the rows are representing the tuples of the table.

54. What is Cardinality?

Ans:

The *number of tuples* in a relation is called the Cardinality..

55. What is Relational Algebra?

Ans:

It is procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

56. What is Relational Calculus?

Ans:

It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. e.g. of languages bases on it are DSL ALPHA, QUEL.

57. What is Codd’s 12 Rules?

Ans:

**Codd's 12 rules** are a set of thirteen rules (numbered zero to twelve) proposed by [Edgar F. Codd](http://en.wikipedia.org/wiki/Edgar_F._Codd), a pioneer of the [relational model](http://en.wikipedia.org/wiki/Relational_model) for [databases](http://en.wikipedia.org/wiki/Database), designed to define what is required from a [database management system](http://en.wikipedia.org/wiki/Database_management_system) in order for it to be considered *relational*, i.e., an [RDBMS](http://en.wikipedia.org/wiki/RDBMS). They are sometimes jokingly referred to as "Codd's Twelve Commandments".

**Rule 0:** The system must qualify as [*relational*](http://en.wikipedia.org/wiki/Relational_model), as a [*database*](http://en.wikipedia.org/wiki/Database), and as a [*management system*](http://en.wikipedia.org/wiki/Management_system).

For a system to qualify as a relational database management system ([RDBMS](http://en.wikipedia.org/wiki/RDBMS)), that system must use its *relational* facilities (exclusively) to *manage* the [*database*](http://en.wikipedia.org/wiki/Database).

**Rule 1:** **The *information rule*:**

All information in the database is to be represented in one and only one way, namely by values in column positions within rows of tables.

**Rule 2:** ***The guaranteed access rule*:**

All data must be accessible. This rule is essentially a restatement of the fundamental requirement for [primary keys](http://en.wikipedia.org/wiki/Unique_key). It says that every individual scalar value in the database must be logically addressable by specifying the name of the containing [table](http://en.wikipedia.org/wiki/Table), the name of the containing column and the primary key value of the containing [row](http://en.wikipedia.org/wiki/Row).

**Rule 3:** ***Systematic treatment of null values*:**

The DBMS must allow each field to remain null (or empty). Specifically, it must support a representation of "missing information and inapplicable information" that is [systematic](http://en.wikipedia.org/wiki/Systematic), distinct from all regular values (for example, "distinct from zero or any other number", in the case of numeric values), and independent of [data type](http://en.wikipedia.org/wiki/Data_type). It is also implied that such representations must be manipulated by the DBMS in a systematic way.

**Rule 4:** ***Active*** [***online***](http://en.wikipedia.org/wiki/Online)[***catalog***](http://en.wikipedia.org/wiki/Database_catalog) ***based on the relational model*:**

The system must support an online, inline, relational [catalog](http://en.wikipedia.org/wiki/Database_catalog) that is accessible to authorized users by means of their regular [query language](http://en.wikipedia.org/wiki/Query_language). That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.

**Rule 5:** **The *comprehensive data sublanguage rule*:**

The system must support at least one relational language that

1. Has a [linear syntax](http://en.wikipedia.org/wiki/Linear_syntax)
2. Can be used both interactively and within application programs,
3. Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and [transaction](http://en.wikipedia.org/wiki/Database_transaction) management operations (begin, commit, and rollback).

**Rule 6:** **The *view updating rule*:**

All views that are theoretically updatable must be updatable by the system.

**Rule 7: *High-level insert, update, and delete*:**

The system must support set-at-a-time *insert*, *update*, and *delete* operators. This means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

**Rule 8:** ***Physical data independence:***

Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.

**Rule 9:** ***Logical data independence:***

Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure. Logical data independence is more difficult to achieve than physical data independence.

**Rule 10:** ***Integrity independence*:**

[Integrity constraints](http://en.wikipedia.org/wiki/Integrity_constraints) must be specified separately from application programs and stored in the [catalog](http://en.wikipedia.org/wiki/Database_catalog). It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.

**Rule 11:** ***Distribution independence*:**

The distribution of portions of the database to various locations should be invisible to users of the database. Existing applications should continue to operate successfully:

1. When a distributed version of the DBMS is first introduced; and
2. When existing distributed data are redistributed around the system.

**Rule 12:** **The *nonsubversion rule*:**

If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.

58. What is Constraints?

Ans:

Database constraints are rules that define interrelations between tables and can check and modify the data in a database. These rules are realized as special database objects. The main advantage of using constraints consists in capability to implement data check, and a part of business-logic of the application at a level of a database, i.e. to centralize and simplify it, so to make the development of databases applications easier and more reliable.

Beginning developers often neglect using database constraints, considering that they hamper creative work. However, actually such opinion is formed on insufficient knowledge of the theory and practice of database design.

Constraints

Entity Integrity Referential Integrity

Domain Constraints Key Constraints

Super Key Candidate Key Foreign Key

1) Entity Integrity Constraints 🡪 The constraints that restrict the values of a tuple in a relation is guided by the entity integrity constraints. There are of two types Constraints that guide the entity integrity. They are:

i) Domain Constraints – A domain of possible values should be associated with every attribute. These domain constraints are the most basic form of integrity constraint. They are easy to test for when data is entered.

ii) Key Constraints – Key constraint is implied by the existence of candidate keys. The intension includes a specification of the attribute(s) consisting the primary key and specification of the attribute(s) consisting alternate keys, if any. Each of these specifications implies a uniqueness constraint (by definition of candidate key); in addition primary key specification implies a no-nulls constraint (by integrity rule 1).

2) Referential Integrity constraints 🡪 A referential CONSTRAINT definition (referential\_constraint\_definition) defines an integrity condition that must be satisfied by all the rows in two tables. The resultant dependency between two tables affects changes to the rows contained in them.

A referential CONSTRAINT definition can be used in a CREATE TABLE statement or ALTER TABLE statement (ADD definition). The table (table\_name) specified in the corresponding statement is referred to in the following sections as the referencing table.

The name of a referential constraint can be specified after the keywords FOREIGN KEY.

* If the name of a referential constraint is specified, it must be different from all other names of referential constraints for the referencing table.
* If no referential constraint name is specified, the database system assigns a unique name (based on the referencing table).

59. What is View?

Ans:

In database theory, a **view** consists of a stored query accessible as a ***virtual table*** composed of the result set of a query. Unlike ordinary tables (base tables) in a relational database, a view does not form part of the physical schema: it is a dynamic, virtual table computed or collated fromdata in the database. Changing the data in a table alters the data shown in subsequent invocations of the view.

Views can provide advantages over tables:

* Views can represent a subset of the data contained in a table.
* Views can join and simplify multiple tables into a single virtual table.
* Views can act as aggregated tables, where the database engine aggregates data (sum, average etc) and presents the calculated results as part of the data.
* Views can hide the complexity of data; for example a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table.
* Views take very little space to store; the database contains only the definition of a view, not a copy of all the data it presents.
* Depending on the SQL engine used, views can provide extra security.
* Views can limit the degree of exposure of a table or tables to the outer world.

60. How does Tuple-oriented Relational Calculus differ from Domain-Oriented Relational Calculus?

Ans:

The tuple-oriented relational calculus uses a tuple variables i.e., variable whose only permitted values are tuples of that relation. e.g. QUEL

The Domain-Oriented Calculus Has domain variables i.e., variables that range over the underlying domains instead of over relation. e.g. ILL, DEDUCE.

**61. What is RDBMS KERNEL?**

**Ans:**

Two important pieces of RDBMS architecture are the ***Kernel,*** which is the software, and the data dictionary, which consists of the system-level data structures used by the Kernel to manage the database. You might think of an RDBMS ***as an Operating System*** (or set of Subsystems), designed specifically for ***controlling data access;*** its primary functions are storing, retrieving, and securing data. An RDBMS maintains its own list of authorized users and their associated privileges; manages memory caches and paging; controls locking for concurrent resource usage; dispatches and schedules user requests; and manages space usage within its table-space structures.

**62. Name the Sub-systems of a RDBMS.**

**Ans:**

I/O, Security, Language Processing, Process Control, Storage Management, Logging and Recovery, Distribution Control, Transaction Control, Memory Management, Lock Management.

**63. Which part of the RDBMS takes care of the data dictionary?**

**Ans:**

Data Dictionary is a set of tables and database objects that is stored in a special area of the databases and maintained exclusively by the ***Kernel.***

**64. What is the job of the Information Stored in Data-Dictionary?**

**Ans:**

The information in the Data-Dictionary validates the existence of the objects, provides access to them, and maps the actual Physical Storage location.

**65. Not only RDBMS takes care of locating data it also determines an optimal access path to store or retrieve the data. How do you communicate with an RDBMS?**

**Ans:**

You communicate with an RDBMS using ***Structured Query Language (SQL).***

66. What is ROWID?

Ans:

The ROWID is a unique Database-wide Physical Address for every row on every table. Once assigned (when the row is first inserted into the Database), it never changes until the row is deleted or the table is dropped.

The ROWID consists of the following *three* components, the combination of which uniquely identifies the Physical Storage location of the row.

Oracle Database files number, which contains the block with the rows.

Oracle Block address, which contains the row.

The row within the block (because each block can hold many rows)

The ROWID is used internally in indexes as a quick means of retrieving rows with a particular key value. Application developers also use it in SQL statement as a quick way to access a row once they know the ROWID.

**UNIT-II**

**67. What is Query Language?**

**Ans:**

A query language is a language in which user requests information from the database. Query language can be categorized as either ***procedural*** or ***nonprocedural.***

**68. What is Procedural Query Language?**

**Ans:**

In procedural language, the user instruct the system to do a sequence of operations on database to compute the desired result.

**69. What is Nonprocedural Query Language?**

**Ans:**

In nonprocedural language, the user describes the desired information without giving a specific procedure for obtaining that information.

**70. Describe the fundamental operations of relational algebra.**

**Ans:**

The relational algebra is divided into two broad categories:

1. **Set Orientation operation**
2. **Relation Orientation operation**

**Relational Algebra**

**borrower (cutomer-name, loan-number)**

**depositor (customer-name, account-number)**

**customer (customer-name, number-street, customer-city)**

**71. Write down the relational algebra to find the names of all customer who have either an account or a loan or both.**

**Ans:**

πcustomer-name (borrower) πcustomer-name (depositor)

**72. Write a relational algebra to find the names of all customers who have an account but not a loan.**

**Ans:**

πcustomer-name (depositor) – πcustomer-name (borrower)

**73. Write a relational algebra to find the names of all customers who have loan in ‘Perryridge’ branch.**

**Ans:**

πcutomer-name (σborrower.loan-number = loan.loan-number

(σbranch-name = “Perryridge” (borrower X loan)))

**74. Write a relational algebra that produces the names of all customers who have both a loan and an account.**

**Ans:**

πcustomer-name (borrower) πcustomer-name (depositor)

**75. Write a relational algebra that produce the name of all customers who have both a loan and an account at bank.**

**Ans:**

πcustomer-name (borrower  depositor)

**76. What is Division Operator?**

**Ans:**

The division operator is denoted by ÷, it is suited fro queries that include the phrase ***“for all”.*** The relation A ÷ B is a relation on schema A – B contains all attributes of schema A that are not present in schema B. A tuple is in A÷B iff both the following conditions holds.

* + 1. Tuple (t) is in ПA-B (R)
    2. For every tuple tB­ in B, there is a tuple tA ­in A satisfies both the following conditions.

tA [B] = tB [B]

tA [A – B] = t

**Example:**

P: Q: P ÷ Q:

|  |  |
| --- | --- |
| A | B |
| a1 | b1 |
| a1 | b2 |
| a2 | b1 |
| a3 | b1 |
| a4 | b2 |
| a5 | b1 |
| a5 | b2 |

|  |
| --- |
| B |
| b1 |
| b2 |

|  |
| --- |
| A |
| a1 |
| a5 |

**Relational Algebra**

**EMPLOYEES (emp-no, name, address)**

**PROJECT (p-no, p-name)**

**WORK-ON (emp-no, p-no)**

**PART (part-no, part-name, qty-on-hand, size)**

**USE (emp-no, p-no, number)**

**77. Write the relational algebra to print the names of the employees who are not working in any project.**

**Ans:**

Πname (EMPLOYEE) – Пname (EMPLOYEE  WORK-ON)

**78. Write the relational algebra to list te names of the projects such that every employee working on these project has used a part ‘bolt’ with size 6.**

**Ans:**

A= Пemp-no, p-no σpart-name = “bolt” ^ size = 6 (PART part.part-no = use.p-no USE)

B= WORK-ON  Пp-no (A)

C= B – A

D= Пp-no (A) – Пp-no (C)

N= Пp-name (PROJECT  D)

**79. Write the relational algebra to found the total number of employees in each department.**

**Ans:**

p-noGcount(emp-no)(WORK-ON)

**80. Using tuple calculus find the loan-number for each loan of an amount over $1200.**

**Ans:**

{t / s loan (t[loan\_number]=s[loan\_number] ^ s[amount] > 1200)}

**81. Using tuple calculus find the names of all customers having a loan, an account, or both at the bank.**

**Ans:**

{t / s  borrower (t[customer­\_name]=s[customer­\_name])  u  depositor(t[customer\_name] = u[customer\_name])}

**82. Using tuple calculus find the names of all customers who have a loan at the Perryridge branch, but no account at any branch of the bank.**

**Ans:**

{t / s borrower (t [customer\_name ] = s [customer\_name ]

u loan (u [branch\_name ] = “Perryridge”

u [loan\_number ] = s [loan\_number ]))

**not** v depositor (v [customer\_name ] = t [customer\_name ])}

**83. Using tuple calculus find the names of all customers who have an account at all branches located in Brooklyn.**

**Ans:**

{t / r customer (t [customer\_name ] = r [customer\_name ]) 

( u branch (u [branch\_city ] = “Brooklyn” 

s depositor (t [customer\_name ] = s [customer\_name ]

w account ( w[account\_number ] = s [account\_number ]

( w [branch\_name ] = u [branch\_name ]))))}

**84. What is the job of the Information Stored in Data-Dictionary?**

**Ans:**

The information in the Data-Dictionary validates the existence of the objects, provides access to them, and maps the actual Physical Storage location.

**85. Not only RDBMS takes care of locating data it also determines an optimal access path to store or retrieve the data. How do you communicate with an RDBMS?**

**Ans:**

You communicate with an RDBMS using Structured Query Language (SQL).

SQL & INTEGRITY CONSTRAINTS

86. What is DDL (Data Definition Language)?

Ans:

A data base schema is specifies by a set of definitions expressed by a special language called DDL. The SQL DDL allows specification of not only a set of relations, but also information about each relation, including,

*•* The schema for each relation

*•* The domain of values associated with each attribute

*•* The integrity constraints

*•* The set of indices to be maintained for each relation

*•* The security and authorization information for each relation

*•* The physical storage structure of each relation on disk

Some examples:

* CREATE - To create objects in the database
* ALTER - Alters the structure of the database
* DROP - Delete objects from the database
* TRUNCATE - Remove all records from a table, including all spaces allocated for the records are removed
* COMMENT - Add comments to the data dictionary
* RENAME - Rename an object

87. What is DML (Data Manipulation Language)?

Ans:

The language that enable user to *access or manipulate data* as organized by appropriate data model.

Procedural DML or Low Level (DML requires a user to specify what data are needed and how to get those data.

Non-Procedural DML or High Level (DML requires a user to specify what data are needed without specifying how to get those data.

Some examples:

* SELECT - Retrieve data from the a database
* INSERT - Insert data into a table
* UPDATE - Updates existing data within a table
* DELETE - Deletes all records from a table, the space for the records remain
* MERGE - UPSERT operation (insert or update)
* CALL - Call a PL/SQL or Java subprogram
* EXPLAIN PLAN - Explain access path to data
* LOCK TABLE - Control concurrency

88. What is DCL (Data Control Language)?

Ans:

It allows to control access to the Database.

Some examples:

* GRANT - Gives user's access privileges to database
* REVOKE - Withdraw access privileges given with the GRANT command

89. What is VDL (View Definition Language)?

Ans:

It specifies user views and their mappings to the conceptual schema.

90. What is SDL (Storage Definition Language)?

Ans:

This language is to specify the internal schema. This language may specify the mapping between two schemas.

91. What is Data Storage-Definition Language?

Ans:

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called Data Storage-Definition Language.

92. What is DML Compiler?

Ans:

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

93. What is Query Evaluation Engine?

Ans:

It executes low-level instruction generated by compiler.

94. What is Metadata?

Ans:

Metadata is the *“Data about other Data”.* In other word Metadata describes how and when and by whom a particular set of data was collected, and how the data is formatted.

95. What is DDL Interpreter?

Ans:

It interprets DDL statements and records them in tables containing metadata.

96. What is Record-at-a-time?

Ans:

The Low Level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.

97. What is Set-at-a-time or Set-oriented?

Ans:

The High Level or Non-Procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be Set-at-a-time or Set-Oriented.

98. State different types of domain in SQL.

Ans:

The SQL standard supports a variety of built-in domain types, including:

* **char** (*n*): A fixed-length character string with user-specified length *n*. The full form, **character**, can be used instead.
* **varchar** (*n*): A variable-length character string with user-specified maximum length *n*. The full form, **character varying**, is equivalent.
* **int**: An integer (a finite subset of the integers that is machine dependent). The full form, **integer**, is equivalent.
* **smallint**: A small integer (a machine-dependent subset of the integer domain type).
* **numeric** (*p, d*): A fixed-point number with user-specified precision. The number consists of *p* digits (plus a sign), and *d* of the *p* digits are to the right of the decimal point. Thus, **numeric** (3, 1) allows 44*.*5 to be stored exactly, but neither 444*.*5 nor 0*.*32 can be stored exactly in a field of this type.
* **real, double precision**: Floating-point and double-precision floating-point numbers with machine-dependent precision.
* **float** (*n*): A floating-point number, with precision of at least *n* digits.
* **date**: A calendar date containing a (four-digit) year, month, and day of the month.
* **time**: The time of day, in hours, minutes, and seconds. A variant, **time** (*p*), can be used to specify the number of fractional digits for seconds (the default being 0). It is also possible to store time zone information along with the time.
* **timestamp**: A combination of **date** and **time**. A variant, **timestamp** (*p*), can be used to specify the number of fractional digits for seconds (the default here being 6).

**99. What is Null Value in SQL?**

**Ans:**

Null values (distinct from an empty character string or a string of blank characters and distinct from zero or any other number) are supported in a fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of the data type.

Null Value is introduced by the creator of the relational database model, E. F. Codd.

100. What is the purpose of creating the Views in DBMS?

Ans:

A view is generally used to provide a ***subset view of a table.*** This can be for ***security or clarity.***

For example, you have a payroll table like so:

name, address, city, state, zip, phone, supervisor, salary, last\_paycheck\_date, no\_demerits, notes, and you only want to have an employee see name, address, phone number and supervisor, then they can't see his information. Similarly, if you have multiple tables and need only 5 or 6 fields, then you can use a view to select only those fields.

A view also has the advantage that it uses less resources on the SQL server side because the SQL code is already compiled, and (in some cases), the SQL server keeps the data references up-to-date, so as not to have to find them every time it's run.

**101. What is a Query?**

**Ans:**

A Query with respect to DBMS relates to user commands that are used to interact with a Database. The Query Language can be classified in Data Definition Language and Data Manipulation Language.

**102. What do you mean by Correlated Subquery?**

**Ans:**

Subqueries or Nested Queries are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is a called a Correlated Subquery.

A Correlated Subquery can be easily identified if it contains any references to the Parent Subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a Non-Correlated Subquery.

e.g. Select \* From CUST

Where ‘10/03/1990’

IN (Select ODATE From ORDER

Where CUST.CNUM = ORDER.CNUM)

**103. What are the Primitive Operations common to all Record Management Systems?**

**Ans:**

Addition, Deletion and Modification.

**104. In which buffer all the commands that are typed in are stored?**

**Ans:**

‘Edit’ Buffer

**105. What are the Unary Operations in Relational Algebra?**

**Ans:**

PROJECTION and SELECTION.

**106. Are the resulting relations of PRODUCT and JOIN Operation same?**

**Ans:**

PRODUCT: Concatenation of every row in one relation with every row in another.

JOIN: Concatenation of rows from one relation and related rows from another.

107. Which is the subset of SQL commands used to manipulate Oracle Database Structures, including tables?

Ans:

Data Definition Language (DDL)

108. What Operator performs Pattern Matching?

Ans:

LIKE Operator

109. What Operator tests column for the absence of Data?

Ans:

IS NULL Operator

110. Which command executes the contents of a specified file?

Ans:

START <filename> or @ <filename>

111. What is the Parameter substitution symbol used with INSERT INTO command?

Ans: &

112. Which command displays the SQL command in the SQL Buffer, and then executes it?

Ans:

RUN

113. What are the Wildcards used for Pattern Matching?

Ans:

\_ for single character substitution and % for multi-character substitution

114. State true or False. EXISTS, SOME, ANY are operators in SQL.

Ans:

True

115. State True or False. !=, <>, ^= all denote the same operation.

Ans:

True.

116. What are the Privileges that can be granted on a table by a user to others?

Ans:

Insert, Update, Delete, Select, References, Index, Execute, Alter, All.

117. What command is used to get back the privileges offered by the GRANT command?

Ans:

REVOKE

118. Which system tables contain information on privileges granted and privileges obtained?

Ans:

USER\_TAB\_PRIVS\_MADE, USER\_TAB\_PRIVS\_RECD

119. Which system table contains information on constraints on all the tables created?

Ans:

USER\_CONSTRAINTS

120. What is the output of DELETE and TRUNCATE commands?

Ans:

Both will result in deleting all the rows in the table.

121. What is the difference between TRUNCATE and DELETE command?

Ans:

TRUNCATE is a DDL command whereas DELETE is a DML command. Hence DELETE operation *can be Rolled Back,* but TRUNCATE operation *cannot be Rolled Back.* *WHERE* clause *can be used* with DELETE and *not* with TRUNCATE.

122. What command is used to create a table by copying the structure of another table?

Ans:

CREATE TABLE <table name> AS SELECT <command>

Explanation:

To copy only the structure, the WHERE clause of the SELECT command should contain a FALSE statement as in the following.

CREATE TABLE NEWTABLE AS SELECT \* FROM

EXISTINGTABLE WHERE 1 = 2;

If the WHERE condition is True, then all the rows or rows satisfying the condition will be copied to the new table.

123. What will be the output of the following query?

SELECT REPLACE (TRANSLATE (LTRIM (RTRIM (‘!! ATHEN !!’, ‘!’), ‘!’), ‘AN’, ‘\*\*’), ‘\*’, ‘TROUBLE’) FROM DUAL;

Ans:

TROUBLETHETROUBLE

124. What will be the output of the following Query?

SELECT DECODE (TRANSLATE (‘A’, ‘1234567890’, ‘1111111111’), ‘1’, ‘YES’, ‘NO’);

Ans:

NO.

Because, the Query checks whether a given String is a numerical digit.

125. What does the following Query do?

SELECT SAL + NVL (COMM, 0) FROM EMP;

Ans:

This displays the total Salary of all Employees. The Null Values in the commission column will be replaced by 0 and added to Salary.

126. Which Date Function is used to find the difference between two dates?

Ans:

MONTHS\_BETWEEN

127. Why does the following command give a compilation error?

DROP TABLE & TABLE-NAME;

Ans:  
Variable names should start with an alphabet. Here the table name starts with an ‘&’ symbol.

128. What is the Advantage of specifying WITH GRANT OPTION in the GRANT command?

Ans:

The privilege receiver can further grant the privilege he/she has obtained from the owner to any other user.

129. What is the use of the DROP option in the ALTER TABLE command?

Ans:

It is used to DROP constraints specified on the table.

130. What is the value of ‘COMM’ and ‘SAL’ after executing the following query if the initial value of ‘SAL’ is 10000?

UPDATE EMP SET SAL = SAL + 1000, COMM = SAL \* 0.1;

Ans:

SAL = 11000, COMM = 1000

131. What is the use of DESC in SQL?

Ans:

DESC has two purposes. It is used to describe a schema as well as to retrieve rows from table in descending order.

Explanation:

The query SELECT \* FROM ORDER BY ENAME DESC will display the output sorted on ENAME in descending order.

132. What is the use of CASCADE CONSTRAINTS?

Ans:

When this clause is used with the DROP command, a parent table can be dropped even when a child table exists.

133. Which function is used to find the largest integer less than or equal to a specific value?

Ans:

FLOOR

134. What is the output of the following query?

SELECT TRUNC (1234.5678, -2) FROM DUAL;

Ans:

1200

Quick Review : SQL

* The most important DDL statements in SQL.

CREATE TABLE – creates a new database table.

ALTER TABLE – alters (changes) a database table.

DROP TABLE – deletes a database table.

CREATE INDEX – creates an index (search key)

DROP INDEX – deletes an index.

* Operators used in SELECT statements.

= Equal

<> or != Not Equal

> Greater than

< Less than

>= Greater than or Equal

<= Less than or Equal

BETWEEN Between an inclusive range

LIKE Search for a pattern

* SELECT statements:

SELECT column\_name(s) FROM table\_name

SELECT DISTINCT column\_name(s) FROM table\_name

SELECT column FROM table WHERE column operator value

SELECT column FROM table WHERE column LIKE pattern

SELECT column, SUM (column) FROM table GROUP BY column

SELECT column, SUM (column) FROM table GROUP BY column HAVING SUM (column) condition value

Note that single quotes around text values and numeric values should not be enclosed in quotes. Double quotes may be acceptable in some databases.

* The SELECT INTO Statement is most often used to Create Backup Copies of table or fro archiving records.

SELECT column\_name(s) INTO new\_table [IN external\_database]

FROM

source WHERE column\_name operator value

* The INSERT INTO Statements.

INSERT INTO table\_name VALUES (value1, value2, ….)

INSERT INTO table\_name (column1, column2, ….) VALUES (value1, value2, ….)

* The UPDATE statement:

UPDATE table\_name SET column\_name = new\_value WHERE column\_name = some\_value

* The DELETE Statements:

DELETE FROM table\_name WHERE column\_name = some\_value

DELETE All Rows:

DELETE FROM table\_name

Or,

DELETE \* FROM table\_name

* Sort the Rows:

SELECT column1, column2, … FROM table\_name ORDER BY

columnX,

columnY, …

SELECT column1, column2, … FROM table\_name ORDER BY

ColumnX DESC

SELECT column1, column2, … FROM table\_name ORDER BY

columnX DESC,

columnY ASC

* The IN operator may be used if you know the exact value you want to return for atleast one of the columns.

SELECT column\_name FROM table\_name WHERE column\_name IN (value1, value2, …)

* BETWEEN … AND

SELECT column\_name FROM table\_name WHERE column\_name

BETWEEN value1

AND value2

Note – The value can be numbers, text or dates.

* What are the privileges that can be granted on a table by a user to others?

Insert , uodate, delete, select, references, index, execute, alter, all.

QUERIES

Table 1: STUDIES

PNAME (VARCHAR), SPLACE (VARCHAR), COURSE (VARCHAR), CCOST (NUMBER)

Table 2: SOFTWARE

PNAME (VARCHAR), TITLE (VARCHAR), DEVIN (VARCHAR), SCOST (NUMBER), DCOST (NUMBER), SOLD (NUMBER)

Table 3: PROGRAMMER

PNAME (VARCHAR), DOB (DATE), DOJ (DATE), SEX (VARCHAR), PROF1 (VARCHAR), PROF2 (VARCHAR), SAL (NUMBER)

LEGEND:

PNAME – Programmer Name, SPLACE – Study Place, CCOST – Course Cost, DEVIN – Developed in, SCOST –Software Cost, DCOST – Development Cost, PROF1 – Proficiency 1, PROF2 – Proficiency 2, SAL – Salary.

135. Find out the selling cost average for packages developed in Oracle.

Ans:

SELECT AVG (SCOST) FROM SOFTWARE WHERE DEVIN = ‘ORACLE’;

136. Display the names, ages and experience of all programmers.

Ans:

SELECT PNAME, TRUNC (MONTHS\_BETWEEN (SYSDATE, DOB)/12) “AGE” TRUNC (MONTHS\_BETWEEN (SYSDATE, DOJ)/12) “EXPERIENCE” FROM PROGRAMMER;

137. Display the names of those who have done the PGDCA course.

Ans:

SELECT PNAME FROM STUDIES WHERE COURSE = ‘PGDCA’;

138. What is the highest number of copies sold by a package?

Ans:

SELECT MAX (SOLD) FROM SOFTWARE;

139. Display the names and date of birth of all programmers born in April.

Ans:

SELECT PNAME, DOB FROM PROGRAMMER WHERE DOB LIKE ‘%APR%’;

140. Display the lowest course fee.

Ans:

SELECT MIN (CCOST) FROM STUDIES;

141. How many programmers have done the DCA course?

Ans:

SELECT COUNT \* FROM STUDIES WHERE COURSE = ‘DCA’;

142. How much revenue has been earned through the sale of packages developed in C?

Ans:

SELECT SUM (SCOST\*SOLD-DCOST) FROM SOFTWARE GROUP BY DEVIN HAVING DEVIN = ‘C’;

143. Display the details of software developed by Rakesh.

Ans:

SELECT \* FROM SOFTWARE WHERE PNAME = ‘RAKESH’;

144. How many programmers studies at Pentafour?

Ans:

SELECT \* FROM STUDIES WHERE SPLACE = ‘PENTAFOUR’;

145. Display the details of packages whose sales crossed the 500 mark.

Ans:  
SELECT \* FROM SOFTWARE WHERE SCOST\*SOLD-DCOST > 5000;

146. Find out the number of copies which should be sold in order to recover the development cost of each package.

Ans:

SELECT CEIL (DCOST/SCOST) FROM SOFTWARE.

147. Display the details of packages for which the development cost has been recovered.

Ans:

SELECT \* FROM SOFTWARE WHERE SCOST\*SOLD >= DCOST;

148. What is the price of costliest software developed in VB?

Ans:

SELECT MAX (SCOST) FROM SOFTWARE GROUP BY DEVIN HAVING DEVIN = ‘VB’;

149. How many packages were developed in Oracle?

Ans:

SELECT COUNT (\*) FROM SOFTWARE WHERE DEVIN = ‘ORACLE’;

150. How many programmers studies at PRAGATHI?

Ans:

SELECT COUNT \* FROM STUDIES WHERE SPLACE = ‘PRAGATHI’;

151. How many programmers paid 10000 to 15000 for the course?

Ans:

SELECT COUNT \* FROM STUDIES WHERE CCOST BETWEEN 10000 AND 15000;

152. What is the average course fees?

Ans:

SELECT AVG (CCOST) FROM STUDIES;

153. Display the details of programmers knowing C.

Ans:

SELECT \* FROM PROGRAMMER WHERE PROF1=‘C’ OR PROF2=‘C’;

154. How many programmers know either C or Pascal?

Ans:

SELECT \* FROM PROGRAMMER WHERE PROF1 IN (‘C’, ‘PASCAL’) OR PROF2 IN (‘C’, ‘PASCAL’);

155. How many programmers don’t know C and C++?

Ans:

SELECT \* FROM PROGRAMMER WHERE PROF1 NOT IN (‘C’, ‘C++’) AND PROF2 NOT IN (‘C’, ‘C++’);

156. How old is the oldest male programmer?

Ans:

SELECT TRUNC (MAX (MONTHS\_BETWEEN (SYSDATE, DOB)/12)) FROM PROGRAMMER WHERE SEX = ‘M’;

157. What is the average age of female programmers?

Ans:

SELECT TRUNC (AVG (MONTH\_BETWEEN (SYSDATE, DOB)/12)) FROM PROGRAMMER WHERE SEX = ‘F’;

158. Calculate the experience in years for each programmer and display along with their names in descending order.

Ans:

SELECT PNAME, TRUNC (MONTHS\_BETWEEN (SYSDATE, DOJ)/12) FROM PROGRAMMER ORDER BY PNAME DESC;

159. Who are the programmers who celebrate their birthdays during the current month?

Ans:

SELECT PNAME FROM PROGRAMMER WHERE TO\_CHAR (DOB, ‘MON’) = TO\_CHAR (SYSDATE, ‘MON’);

160. How many female programmers are there?

Ans:

SELECT COUNT \* FROM PROGRAMMERWHERE SEX = ‘F’;

161. What are the languages known by the male programmer?

Ans:

SELECT DISTINCT (PROF1) FROM PROGRAMMER WHERE SEX = ‘M’;

162. What is the average salary?

Ans:

SELECT AVG (SAL) FROM PROGRAMMER;

163. How many people draw 5000 to 7500?

Ans:

SELECT COUNT \* FROM PROGRAMMER WHERE SAL BETWEEN 5000 AND 7500;

164. Display the details of those who don’t know C, C++ or Pascal.

Ans:

SELECT \* FROM PROGRAMMER WHERE PROF1 NOT IN (‘C’, ‘C++’, ‘PASCAL’) AND PROF2 NOT IN (‘C’, ‘C++’, ‘PASCAL’);

165. Display the costliest package developed by each programmer.

Ans:

SELECT PNAME, TITLE, SCOST FROM SOFTWARE WHERE SCOST IN (SELECT MAX(SCOST) FROM SOFTWARE GROUP BY PNAME);

165. Produce the following output for all the male programmers

Ans:

SELECT ‘Mr.’ || PNAME || ‘-has’ ||

TRUNC (MONTHS\_BETWEEN (SYSDATE, DOJ)/12) || ‘years of experience’ “Programmer” FROM PROGRAMMER WHERE SEX = ‘M’ UNION SELECT ‘Ms.’ || PNAME || ‘-has’ || TRUNC (MONTHS\_BETWEEN (SYSDATE, DOJ)/12) || ‘years of experience’ “Programmer” FROM PROGRAMMER WHERE SEX = ‘F’;

QUERIES

Table 1: DEPT

DEPTNO (NOT NULL, NUMBER (2)), DNAME (VARCHAR2(14)), LOC (VARCHAR2(13))

Table2: EMP

EMPNO (NOT NULL, NUMBER(4)), ENAME (VARCHAR2(10)), JOB (VARCHAR2(9)), MGR (NUMBER(4)), HIREDATE (DATE), SAL (NUMBER(7,2)), DEPTNO(NUMBER(2))

*MGR is the EMPNO of the employee whom the employee reports to. DEPTNO is a foreign key*.

166. List all the employees who have at least one person reporting to them.

Ans:

SELECT DISTINCT (A.ENAME) FROM EMP A, EMP B WHERE A.EMPNO = B.MGR;

OR,

SELECT ENAME FROM EMP WHERE EMPNO IN (SELECT MGR FROM EMP);

167. List the employee details if and if more than 10 employees are present in department no 10.

Ans:

SELECT \* FROM EMP WHERE DEPTNO IN (SELECT DEPTNO FROM EMP GROUP BY DEPTNO HAVING COUNT (EMPNO)>10 AND DEPTNO =10);

168. List the employee details whose name is started and end by ‘A’.

Ans:

SELECT \* FROM EMP WHERE ENAME LIKE “A[A-Z]\*A”;

169. List the name of the employees with their immediate higher authority.

Ans:

SELECT A.ENAME “EMPLOYEE”, B.ENAME “REPORTS TO” FROM EMP A, EMP B WHERE A.MGR = B.EMPNO;

170. List the employee details whose salary is greater than the lowest salary of an employee belonging to DEPTNO 20.

Ans:

SELECT \* FROM EMP WHERE SAL > (SELECT MIN (SAL) FROM EMP GROUP BY DEPTNO HAVING DEPTNO =20);

171. List the details of the employee earning more than the highest paid manager.

Ans:

SELECT \* FROM EMP WHERE SAL > (SELECT MAX(SAL) FROM EMP GROUP BY JOB HAVING JOB = ‘MANAGER’);

172. List the highest salary paid for each job.

Ans:

SELECT JOB, MAX (SAL) FROM EMP GROUP BY JOB;

173. Find the most recently hired employee in each department.

Ans:

SELECT \* FROM EMP WHERE (DEPT.NO, HIREDATE) IN (SELECT DEPTNO, MAX(HIREDATE) FROM EMP GROUP BY DEPTNO);

174. In which year did most people join the company? Display the year and the number of employees.

Ans:

SELECT TO\_CHAR (HIREDATE, ‘YYYY’) “YEAR”, COUNT (EMPNO) “NO. OF EMPLOYEES” FROM EMP GROUP BY

TO\_CHAR (HIREDATE, ‘YYYY’) HAVING COUNT (EMPNO) =

(SELECT MAX (COUNT (EMPNO)) FROM EMP GROUP BY TO\_CHAR (HIREDATE, ‘YYYY’));

175. Which department has the highest annual remuneration bill?

Ans:

SELECT DEPTNO, LPAD (SUM (12\* (SAL + NVL (COMM, 0))), 15) “COMPENSATION” FROM EMP GROUP BU DEPTNO HAVING SUM (12\*(SAL + NVL (COMM, 0))) =

(SELECT MAX(SUM (12\*(SAL + NVL (COMM, 0)))) FROM GROUP BY DEPTNO);

176. Write a query to display a ‘\*’ against the row of the most recently hired employee.

Ans:

SELECT ENAME, HIREDATE, LPAD (‘\*’, “RECENTLY HIRED” FROM EMP WHERE HIREDATE = (SELECT MAX (HREDATE) FROM EMP) UNION SELECT ENAME NAME, HIREDATE, LPAD (‘’, 15) “RECENTLY HIRED” FROM EMP WHERE HIREDATE != (SELECT MAX (HIREDATE) FROM EMP);

177. Write a correlated sub-query to list out the employees who earn more than the average salary of their department.

Ans:

SELECT ENAME, SAL FROM EMP E WHERE SAL > (SELECT AVG (SAL) FROM EMP F WHERE E.DEPTNO =F.DEPTNO);

178. Find the nth maximum salary.

Ans:

SELECT ENAME, SAL FROM A WHERE &N = (SELECT COUNT (DISTINCT (SAL)) FROM EMP B WHERE A.SAL <= B.SAL);

179. Select the duplicate records (Records, which are inserted, that already exist) in the EMP table.

Ans:

SELECT \* FROM EMP A WHERE A.EMPNO IN (SELECT EMPNO FROM EMP GROUP BY EMPNO HAVING COUNT (EMPNO) > 1) AND A.ROWID != MIN(ROWID));

180. Write a query to list the length of service of the employees (of the form n years and m months).

Ans:

SELECT ENAME

“EMPLOYEE”, TO\_CHAR (TRUNC (MONTHS\_BETWEEN (SYSDATE, HIREDATE)/12)) || ‘YEARS’ ||

TO\_CHAR (TRUNC (MOD (MONTHS\_BETWEEN (SYSDATE, HIREDATE)/12))) || ‘MONTHS’ “LENGTH OF SERVICE” FROM EMP;

**181. Define SQL and State the differences between SQL and other Conventional Programming Language.**

Ans:

SQL is a *Non-Procedural Language* that is designed specifically for *data access operations* on Normalized Relational Database Structures.

The primary difference between SQL and other Conventional Programming Language is that SQL statements specify what data operations should be performed rather than how to perform them.

182. What is Data Security in DBMS?

Ans:

Data security is the protection of the database from unauthorized users. Only the authorized persons are allowed to access the database. Most of the users are allowed to access a part of database i.e., the data that is related to them or related to their department. Mostly, the DBA or head of department can access all the data in the database. Some users may be permitted only to retrieve data, whereas others are allowed to retrieve as well as to update data. The database access is controlled by the DBA. He creates the-accounts of users and gives rights to access the database. Typically, users or group of users are given usernames protected by passwords.

Most of the DBMSs provide the security sub-system, which the DBA uses to create accounts of users and to specify account restrictions. The user enters his/her account number (or user name) and password to access the data from database.

Security of the database involves the protection of the database against:

* unauthorized disclosures
* alteration
* destruction

The protection which security gives is usually directed against two classes of user

* Stop people without database access from having any form of access.
* Stop people with database access from performing actions on the database which are not required to perform their duties.

183. What are the aspects of data security?

Ans:

There are many aspects to security :-

* **Legal, social and ethical aspects🡪**

Legally there is the Data Protection Act, which places restrictions on databases which contain information on living people. This was created to protect the public from data contained on a computer, about themselves, to which the public had previously no legal right of access. Information on computers can be wrong, and decisions made on wrong information concerns the public and additionally is of no benefit to the company holding the data. The act supports the idea of the public querying data, and indicating errors in that data.

However, just because a database is legal does not make it socially or ethically acceptable. Collating medical records on computer for a hospital is acceptable, but not having enough security to prevent insurance companies accessing the database and using that as a basis for rejecting life assurance applications could be considered questionable. Frequently it is best to place the tightest restrictions on who can access data, and where necessary security is deliberately relaxed to allow only legitimate queries to take place.

* **Physical controls🡪**

Security often begins with physical controls. If a person cannot enter the building where the database runs and is accessed, then that person cannot access the database. Usually the construction of security is a layered approach, where a person bent on accessing the database must penetrate multiple levels of security. The simple precaution of having all the database access points behind locked doors can only add to the security of the system.

* **Policy questions🡪**

Security of a database is often the enforcement in the database of the company policy. All companies should have a policy statement, listing what is acceptable and what is not. Companies with weak policy statements will often have the weakest security. At a minimum, it should be policy that data stored in the database should not be made available to outside agents without written consent from a Managing Director. Without a policy statement, it is hard to argue that an employee has actually done anything wrong.

* **Operational problems🡪**

If only a single person has access to a database, security is certainly higher than if many people have access. However, if all the people in the UK had to phone the same one person to find out what their bank balance was the whole system would quickly become unworkable. Security considerations often have to be balanced against operational issues.

* **Hardware controls🡪**

No matter how secure the database actually is, if a person can simply steal the hard drive on which the database is stored, then that person can access the database at leisure. This case is obvious, but less obvious security failures, such as taking a copy of a backup tape of the database, can be harder to safeguard against.

* **Operating system security🡪**

Most DBMS's run on top of an operating system (OS). Examples of OS's include Window 95, Windows NT, and Unix. The database may be secure from within the DBMS, but if the database can also be accessed from the OS using simple file handling programs, then a clear weakness in the security model exists.

* **Database system security🡪**

Within the DBMS itself, if anyone can access anything then having any other sort of security seems pointless. The use of user accounts and password protection of user identities is a good starting point to improve security. User identities is also an aid to accountability. Protection of certain elements of the database with respect to certain users (or user groups) should always be considered where potentially confidential data is being stored. It is DBMS security which is the focus of this discussion.

184. What is the goal of database security?

Ans:

* Confidentiality: Prevent or detect the unauthorized access to information.
* Integrity: Prevent or detect the unauthorized modification of information.
* Availability: Prevent or detect the unauthorized denial of services

185. What is the difference between DBMS and OS?

Ans:

* Object granularity: DBMS uses a finer granularity (relations, rows, columns, fields) than the OS (files, devices).
* Semantic correlation among data: relations between data pose a threat of security violations through inference.
* Meta-data: A DBMS provides metadata describing relations, attributes, domains, constraints, etc. An OS provides limited or no meta-data.
* Logical and physical objects: An OS only deals with physical objects (files, devices). A DBMS deals with logical objects, independent of OS objects (relations, views).
* Multiple data types: An OS only knows files and read, write and execute permissions. A DBMS has many data types and operations, with separate access modes e.g. for individual access, grouped access, statistical operations, administrative operations, etc.

186. What are the security mechanisms in DBMS?

Ans:

* Different degrees of granularity of access: The DBMS must offer access controls at various degrees of granularity, such as relations, columns, rows or individual data items.
* Different access modes: Typical database access modes are *select*, *insert*, *update*, *delete*. (select means “read”)
* Different types of access controls:
  + name-dependent: depends on name of object.
  + data-dependent: depends on value of object. (can be value in query or value in object.)
  + context-dependent: depends on other objects being accessed, on time, location of user, etc.
* Dynamic authorization: a user’s authorizations can be modified while the database is operational.
* Multilevel protection: The DBMS should support multilevel protection through a mandatory policy.
* Covert channels: The DBMS should be covert channel-free.
* Inference controls: The DBMS should provide a way to assign classifications to aggregate information.
* Polyinstantiation: This mechanism allows the database to have multiple instances of objects, each having their own classification level.
* Auditing: Security-related events should be reported in a structured format such as system journals, audit trails and system logs.
* Flow controls: Check the destination of output obtained through authorized access.
* No back doors: Access to data should occur only via the DBMS.
* Reasonable performance: Security controls should not increase execution times significantly.

187. What are the Integrity Mechanism in DBMS?

Ans:

* Well-formed transactions: Updates may only occur via transactions. (Correct execution is guaranteed via locking)
* Authenticated users: Updates may only be performed by authorized and authenticated users. Authenticating users is typically performed by the OS and need not be duplicated in the DBMS.
* Least privilege: It should be possible to give users the minimum update rights for their task.
* Separation of duties: No single user should be able to corrupt data on his own.
* Continuity of operation: The DBMS should continue to function, without data loss, in case of disasters.
* Reconstruction of events: Improper behavior should be detected (through audit trails).
* Reality checks: This goes beyond the duty of the DBMS. But through some constraints some “impossible” data can be avoided.
* Ease of safe use: Security procedures should be user-friendly, known and fault-free.
* Delegation of authority: The DBMS should support ways to assign privileges according to mandatory or discretionary policies. Typically the SQL *grant/revoke* statements are used to delegate authority.

188. How Security is implemented in DBMS?

Ans:

**• Non Computer-Based Controls 🡪** Concerned with matters such as policies, agreements, and other administrative controls such as physical controls to buildings and safeguarding equipments.

1. Security policy and contingency plans

2. Personnel controls

3. Securing positioning of equipment

4. Secure data and software

5. Escrow agreements

6. Maintenance agreements

7. Physical access control

8. Building Controls

9. Emergency arrangements

**• Computer-Based Controls 🡪** Concerned with electronic administrative

Procedures

1. Authorization & Authentication

2. Access controls

3. Views

4. Backup and recovery

5. Integrity

6. Encryption

7. RAID technology

189. What is Authorization & Authentication?

Ans:

• Authorization is the ***granting of a right or privilege,*** which enables a subject to legitimately have access to a system or a system’s object.

• Authentication is a mechanism that determines whether a user is, who he or she claims to be.

190. What is Access Control?

**Ans:**  
An ***access control system*** is a system which enables an authority to control access to areas and resources in a given physical facility or computer-based information system. An access control system, within the field of physical security, is generally seen as the second layer in the security of a physical structure.

A PIN on an ATM system at a bank is the means of access control.

• The Access Control is based on the granting and revoking of privileges.

• A privilege allows a user to create or access (that is read, write, or modify) some database object (such as a relation, view, and index) or to run certain DBMS utilities.

• Privileges are granted to users to accomplish the tasks required for their jobs.

• Most DBMS provide an approach called Discretionary Access Control (**DAC**).

• SQL standard supports DAC through the GRANT and REVOKE commands.

• The GRANT command gives privileges to users, and the REVOKE command takes away privileges.

• DAC while effective has certain weaknesses. In particular an unauthorized user can trick an authorized user into disclosing sensitive data.

• An additional approach is required called Mandatory Access Control (MAC).

**191. What is View?**

**Ans:**

– The view is the dynamic result of one or more relational operations operating on the base relations to produce another relation.

– A view is a virtual relation that does not actually exist in the database, but is produced upon request by a particular user, at the time of request.

**192. What is Backup & Recovery?**

Ans:

**•** **Backup 🡪** Process of periodically taking a copy of the database and log file (and possibly programs) to offline storage media.

**• Journaling 🡪** Process of keeping and maintaining a log file (or journal) of all changes made to database to enable effective recovery in event of failure.

**• Checkpointing 🡪** Point of synchronization between the database and the transaction log file. All buffers are force-written to secondary storage.

**193. What is Integrity?**

**Ans:**

Prevents data from becoming invalid, and hence giving misleading or incorrect results.

194. What is Triggers?

Ans:

In a DBMS, a ***trigger*** is a SQL procedure that ***initiates an action*** (i.e., *fires* an action) when an event (INSERT, DELETE or UPDATE) occurs. Since triggers are event-driven specialized procedures, they are stored in and managed by the DBMS. A trigger cannot be called or executed; the DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers are ***used to maintain the referential integrity of data*** by changing the data in a systematic fashion.

Each trigger is attached to a single, specified table in the database.

Triggers can be viewed as similar to stored procedures in that both consist of procedural logic that is stored at the database level. Stored procedures, however, are not event-drive and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.

A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a *nested trigger*.

UNIT-III

195. What is Functional Dependency?

Ans: *“A functional dependency is an association between two attributes of the same relation database table. One of the attribute is called the determinant and the other attribute is called the determined. For each value of determinant there is one and only one value of the determined.”*

A Functional Dependency is denoted by X🡪Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then have t1[Y] = t2[Y]. This means that the values of the *Y* component of a tuple in rdepend on, or are determined by,the values of the *X* component; or alternatively, the values of the *X* component of a tuple uniquely (or functionally) determine the values of the *Y* component. We also say that there is a functional dependency from *X* to *Y* or that *Y* is functionally dependent on *X.* The abbreviation for functional dependency is FD or f.d.

196. What is Armstrong’s Axioms?

Ans:

Armstrong's axioms are a set of axioms (or, more precisely, inference rules) used to infer all the functional dependencies on a relational database.

1. Reflexive Rule 🡪

If Y  X, then X  Y

1. Augmentation Rules 🡪

If X  Y, then XZ  YZ

1. Transitive Rules 🡪

If X  Y and Y  Z, then X  Z

1. Decomposition Rules 🡪

If X  YZ then X  Y and X  Z

1. Union Rules 🡪

If X  Y and X  Z, then X  YZ

1. Pseudo Transitive Rules 🡪

If W  X and YX  Z then WY  Z

Of these, the first three are known as *Amstrong Rules.* They are sound because it is enough if a set of FDs satisfy these three. They are called *complete* because using these three rules we can generate the rest all Inference Rules.

197. What is Trivial and Non-trivial Relation?

Ans:

* An FD is trivial if and only if the right side is a subset of the left side.

e.g. XY  Y, this relation is the Trivial Relation.

* Any other dependencies are the Non-trivial Relation.

198. What is Closure?

Ans:

* Closure of a Functional Dependency 🡪

Given a set of dependency F holding on a relation R. Many other dependencies can be logically implemented by F. The set of all those dependencies are called Closure of F and denoted by F+.

e.g. A relation schema R = (A, B, C, D) and the set of functional dependencies are F,

A  B

B  C

A  D

The Functional Dependency A  C is logically implied.

So, this functional dependency is the closure of F.

* Closure of a set of Attribute 🡪

X means a set of attribute that is dependent on the set X; if it contains all the other attributes of that relation then X becomes a Key for the relation. It should be noted that Closure of a set of attribute is a set of attributes whereas Closure of a FDs is a set of FDs.

For a given set of FDs F, F+ (the transitive closure of F) is the set consisting of all the dependencies of F, plus all the dependencies that can be inferred using the Armstrong’s Axioms.

199. What is Equivalence?

Ans:

A set of functional dependencies *E* is covered by a set of functional dependencies *F*—or alternatively, *F* is said to cover *E—*if every FD in *E* is also in *F+*; that is, if every dependency in *E* can be inferred from *F.* Two sets of functional dependencies *E* and *F* are equivalent if E+ = F+. Hence, equivalence means that every FD in *E* can be inferred from *F,* and every FD in *F* can be inferred from *E;* that is, *E* is equivalent to *F* if both the conditions *E* covers *F and F* covers *E* hold.

200. What is the Irreducible set of Dependencies?

Ans:

* A set S of FDs is irreducible if:
  + The right side of every FD in S involves one attribute (a *singleton set*)
  + The left side is *irreducible* (no attribute can be discarded from the determinant without changing the closure S+)
  + No FD in S can be discarded from S without changing the closure of S

For every set of FDs there exists at least one equivalent set that is irreducible.

201. Given the relation R with attributes A, B, C, D and FDs

A 🡪BC, B 🡪C, A 🡪B, AB 🡪C, AC 🡪D

Compute the irreducible set of FDs, which is equivalent to the given set.

Ans:

To start with decompose the FDs to convert the right hand side into singleton

A  B, A  C

B  C, A  B, AB  C, AC  D

The relation A  B holds twice, one of them is dropped.

Now, from A  C, A  AC and from AC  D

By transitivity, we say A  D

Thus the relation AC  D is eliminated.

Now from the relation A  B we say A  AB

Given AB  C, by transitivity A  C

Thus the relation AB  C is eliminated

Thus the irreducible set is:

A  B, B  C, A  D

202. What is Canonical Cover of FDs?

Ans:

A canonical cover for F is a set of dependencies Fc such that

* + F and Fc,are equivalent
  + Fc contains no redundancy
  + Each left side of functional dependency in Fc is unique.

For instance, if we have two FD X→Y, X→Z, we convert them to X→Y∪Z.

203. When is a functional dependency F said to be minimal?

Ans:

Every dependency in F has a single attribute for its right hand side. We cannot replace any dependency X →A in F with a dependency Y→ A that is equivalent to F.

We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

204. What if Fully Functional Dependency?

Ans:

A functional dependency *X* → *Y* is a full functional dependency if removal of any attribute *A* from *X* means that the dependency does not hold any more; that is, for any attribute *A X,* (*X* - {*A*}) does notfunctionally determine *Y.*

205. What is Normalization?

Ans:

It is a process of analyzing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

*Minimizing redundancy, minimizing insertion, deletion and update anomalies.*

206. What is Multivalued Dependency?

Ans:

Multivalued Dependency denoted by X →→Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties

t3[X] = t4[X] = t1[X] = t2[X]

t3[Y] = t1[Y] and t4[Y] = t2[Y]

t3 [Z] = t2 [Z] and t4 [Z] = t1 [Z]

where [Z = (R – (X U Y))]

207. What do you understand by Dependency Preservation?

Ans:

Given a relation R and a set of FDs F, Dependency Preservation states that the Closure of the Union of the Projection of F on each decomposed relation Ri is equal to the Closure of F. i.e.,

((PR1(F)) U … U (PRn(F))+ = F+

If decomposition is not Dependency Preserving, then some Dependency is lost in the Decomposition.

208. Let R (ABCDE) be a relational schema and

F = {AB 🡪 CD, ABC 🡪 E, C 🡪 A} be the set of functional dependencies. Which are the candidate keys of R.

Ans:

Consider the FD, AB 🡪 CD

Here C & D can be derived from A & B. After getting the C the E can be derived from the FD, ABC 🡪 E.

So, AB will be a Candidate Key.

Again, consider the FD, C 🡪 A,

Thus, A can be derived from the C,

Consider the value of B then the E can be derived from the FD, ABC 🡪 E

So, BC will be a Candidate Key.

209. What is 1NF (Normal Form)?

Ans:

First Normal Form (1NF) is now considered to be part of the formal definition of a relation in the basic (flat) relational model; historically, it was defined to disallow multivalued attributes, composite attributes, and their combinations. It states that the domain of an attribute must include only *atomic (simple, indivisible) values* and that the value of any attribute in a tuple must be a *single value* from the domain of that attribute.e.g.

|  |  |  |  |
| --- | --- | --- | --- |
| DNAME | DNUMBER | DMGRSSN | DLOCATIONS |
| Research | 5 | 333445555 | Bellaire, Sugarland, Houston |
| Administration | 4 | 987654321 | Stafford |
| Headquarters | 1 | 888665555 | Houston |

DNAME, DNUMBER and DMGRSSN contain simple value and they are atomic. However, DLOCATIONS is composite. So the relation is not in 1NF.

There are three main techniques to achieve first normal form for such a relation:

1. Remove the attribute DLOCATIONS that violates 1NF and place it in a separate relation DEPT\_LOCATIONS along with the primary key DNUMBER of DEPARTMENT. The primary key of this relation is the combination {DNUMBER, DLOCATION}. A distinct tuple in DEPT\_LOCATIONS exists for *each location* of a department. This decomposes the non-1NF relation into two 1NF relations.
2. Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT, as shown in. In this case, the primary key becomes the combination {DNUMBER, DLOCATION}. This solution has the disadvantage of introducing *redundancy* in the relation.
3. If a *maximum number of values* is known for the attribute—for example, if it is known that *at most three locations* can exist for a department—replace the DLOCATIONS attribute by three atomic attributes: DLOCATION1, DLOCATION2, and DLOCATION3. This solution has the disadvantage of introducing *null values* if most departments have fewer than three locations.

This relation may be modified to be the 1NF is followed:

|  |  |  |  |
| --- | --- | --- | --- |
| DNAME | DNUMBER | DMGRSSN | DLOCATIONS |
| Research | 5 | 333445555 | Bellaire |
| Research | 5 | 333445555 | Sugarland |
| Research | 5 | 333445555 | Houston |
| Administration | 4 | 987654321 | Stafford |
| Headquarters | 1 | 888665555 | Houston |

210. What is 2NF?

Ans:

Second normal form (2NF) is based on the concept of *full functional dependency.*

A relation schema R is in 2NF if it is in 1 NF and every *non-prime attribute (non-key attribute)* A in R is fully functionally dependent on primary key.

The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key. If the primary key contains a single attribute, the test needs not to be applied at all. A relation schema *R* is in 2NF if every nonprime attribute *A* in *R* is *fully functionally dependent* on the primary key of *R.*

e.g.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SSN | PNUMBER | HOURS | ENAME | PNAME | PLOCATION |

In this relation {SSN, PNUMBER} are the Primary Key – Key Attribute and {HOURS, ENAME, PNAME, PLOCATION} are the Non-key Attribute. The Non-key Attributes ENAME, PNAME, PLOCATION are Partially Functional Dependent on the Key Attributes. So this relation is not in 2NF.

To make the relation in 2NF the relation have to be decomposed.

|  |  |  |
| --- | --- | --- |
| SSN | PNUMBER | HOURS |

|  |  |
| --- | --- |
| SSN | ENAME |

|  |  |  |
| --- | --- | --- |
| PNUMBER | PNAME | PLOCATION |

211. What is 3NF?

Ans:

Third normal form (3NF) is based on the concept of *transitive dependency.* A relation schema R is in 3NF if it is in 2NF and for every FD X→A either of the following is true

* X is a Super-key of R.
* A is a Prime Attribute of a R.

According to Codd’s original definition, a relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key.

e.g.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SSN | ENAME | BDATE | DNUMBER | DNAME |

In this relation SSN is the Primary Key directly related to the ENAME, BDATE and DNUMBER. But DNAME relates with SSN through DNUMBER. So this is a Transitive Relation. Thus, this relation is not in 3NF. To make it in 3NF, this relation has to be decomposed.

|  |  |  |  |
| --- | --- | --- | --- |
| SSN | ENAME | BDATE | DNUMBER |

|  |  |
| --- | --- |
| DNUMBER | DNAME |

212. What is BCNF (Boyce-Codd Normal Form)?

Ans:

Boyce-Codd Normal Form (BCNF) was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF, because every relation in BCNF is also in 3NF; however, a relation in 3NF is *not necessarily* in BCNF.

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that of every FD X→A, X must be a superkey—that is, *X* is either a candidate key or a superset thereof.

e.g.

|  |  |  |  |
| --- | --- | --- | --- |
| PROPERTY\_ID# | COUNTY\_NAME | LOT# | AREA |

Here, PROPERTY\_ID#, the Primary Key of this relation, is Functionally Dependent on the other Non-Key Attributes. But some Non-Key Attributes are also Functionally Dependent on the Key Attribute. So, this relation is not in BCNF and has to be decomposed to make it in BCNF.

|  |  |  |
| --- | --- | --- |
| PROPERTY\_ID# | LOT# | AREA |

|  |  |
| --- | --- |
| AREA | COUNTY\_NAME |

213. What is the difference between 3NF and BCNF?

Ans:

The difference between 3NF and BCNF is that for a functional dependency A→B, 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key.

Whereas BCNF insists that for this dependency to remain in a relation, A must be a candidate key.

214 What is 4NF?

Ans:

4NF is concerned with a more general type of dependency known as a *multivalued dependency.* A table is in 4NF if and only if, for every one of its *non-trivial multivalued dependencies* *X →→ Y*, *X* is a superkey—that is, *X* is either a candidate key or a superset thereof.

215. What is 5NF?

Ans:

A Relation schema R is said to be 5NF if for every join dependency {R1, R2, …., Rn} that holds R, one the following is true

Ri = R for some i.

The join dependency is implied by the set of FD, over R in which the left side is key of R.

216. What is Domain-Key Normal Form?

Ans:

A Relation is said to be in DKNF if all constraints and dependencies that should hold on the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

217. What is Join Dependency and Inclusion Dependency?

Ans:

Join Dependency: A Join Dependency is generalization of Multivalued Dependency. A JD {R1, R2, …, Rn} is said to hold over a relation R if R1, R2, R3, …Rn is a lossless-join decomposition of R. There is no set of sound and complete inference rules for JD.

Inclusion Dependency: An Inclusion Dependency is a statement of the form that some columns of a relation are contained in other columns. A Foreign Key constraint is an example of Inclusion Dependency.

218. What is Lossless join property?

Ans:

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

219. What is Decomposition?

Ans:

Let R be a relation schema

A set of relation schemas {R1, R2, …, Rn} is a decomposition of R if

* R = R1 R2  …..  Rn
* each Ri is a subset of R (for i = 1, 2…, n)

220. What is the Goal of Decomposition?

Ans:

* Eliminate redundancy by decomposing a relation into several relations in a higher normal form.
* It is important to check that a decomposition does not lead to bad design

221. What problems may arise due to Decomposition?

Ans:

Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation – information loss

222. What is Lossy & Lossless Decomposition?

Ans:

A decomposition is called Lossy Decomposition or Lossy-join Decomposition if any information *lose* due to decomposition.

A decomposition {R1, R2, …, Rn} of a relation R is called a Lossless Decomposition for R if the natural join of R1, R2, …, Rn produces exactly the relation R.

Let *R* be a relation schema, and let *F* be a set of functional dependencies on *R*.

Let *R*1 and *R*2 form a decomposition of *R*. This decomposition is a lossless-join decomposition of *R* if at least one of the following functional dependencies is in *F+*:

*• R*1 *∩ R*2 *→ R*1

*• R*1 *∩ R*2 *→ R*2

In other words, if *R*1 *∩ R*2 forms a superkey of either *R*1 or *R*2, the decomposition of *R* is a lossless-join decomposition.

223. A given Relation R = (A1, A2, A3, A4, A5) is decomposed into

R1 = (A1, A2, A3, A5), R2 = (A1, A3, A4), R3 = (A4, A5)

Required FDs are,

FD1: A1 → A3 A5, FD2: A5 → A1 A4, FD3: A3 A4 → A2

Show that the decomposition is Lossless Decomposition.

Ans:

Step 1: Fill up this table according to the decomposed relations,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 | A5 |
| FD1 | a(1) | a(2) | a(3) | b(1, 4) | a(5) |
| FD2 | a(1) | b(2, 2) | a(3) | a(4) | b(2, 5) |
| FD3 | b(3, 1) | b(3, 2) | b(3, 3) | a(4) | a(5) |

Step 2: By considering the FD1: A1→A3 A5, we get the new result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 | A5 |
| FD1 | a(1) | a(2) | a(3) | b(1, 4) | a(5) |
| FD2 | a(1) | b(2, 2) | a(3) | a(4) | a(5) |
| FD3 | b(3, 1) | b(3, 2) | b(3, 3) | a(4) | a(5) |

Step 3: By considering the FD2: A5→A1 A4, we get the new result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 | A5 |
| FD1 | a(1) | a(2) | a(3) | a(4) | a(5) |
| FD2 | a(1) | b(2, 2) | a(3) | a(4) | a(5) |
| FD3 | a(1) | b(3, 2) | b(3, 3) | a(4) | a(5) |

As we get the 1st row contains all ‘a’, thus we can say that the decomposition is a Lossless Decomposition.

Unit-IV

**Concurrency Control**

* In the concurrency control, the multiple transactions can be executed simultaneously.
* It may affect the transaction result. It is highly important to maintain the order of execution of those transactions.

## Problems of concurrency control

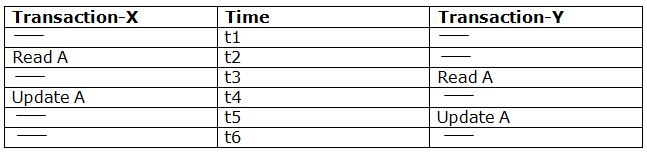
Several problems can occur when concurrent transactions are executed in an uncontrolled manner. Following are the three problems in concurrency control.

1. Lost updates
2. Dirty read
3. Unrepeatable read

### **1. Lost update problem**

* When two transactions that access the same database items contain their operations in a way that makes the value of some database item incorrect, then the lost update problem occurs.
* If two transactions T1 and T2 read a record and then update it, then the effect of updating of the first record will be overwritten by the second update.

**Example:**



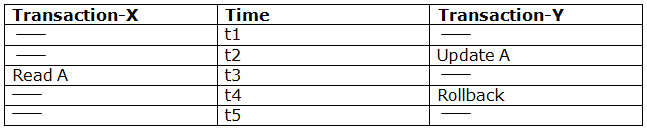
**Here,**

* At time t2, transaction-X reads A's value.
* At time t3, Transaction-Y reads A's value.
* At time t4, Transactions-X writes A's value on the basis of the value seen at time t2.
* At time t5, Transactions-Y writes A's value on the basis of the value seen at time t3.
* So at time T5, the update of Transaction-X is lost because Transaction y overwrites it without looking at its current value.
* Such type of problem is known as Lost Update Problem as update made by one transaction is lost here.

### **2. Dirty Read**

* The dirty read occurs in the case when one transaction updates an item of the database, and then the transaction fails for some reason. The updated database item is accessed by another transaction before it is changed back to the original value.
* A transaction T1 updates a record which is read by T2. If T1 aborts then T2 now has values which have never formed part of the stable database.

**Example:**



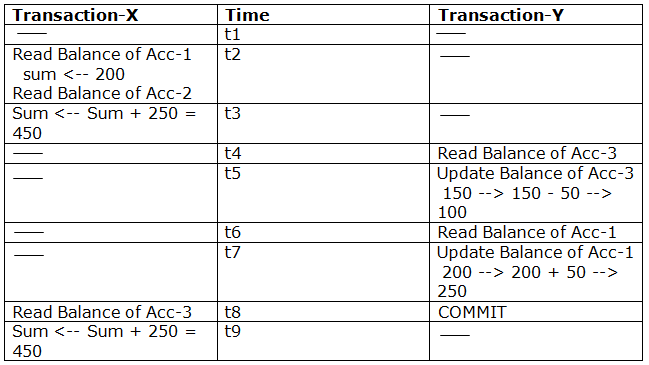
* At time t2, transaction-Y writes A's value.
* At time t3, Transaction-X reads A's value.
* At time t4, Transactions-Y rollbacks. So, it changes A's value back to that of prior to t1.
* So, Transaction-X now contains a value which has never become part of the stable database.
* Such type of problem is known as Dirty Read Problem, as one transaction reads a dirty value which has not been committed.

### **3. Inconsistent Retrievals Problem**

* Inconsistent Retrievals Problem is also known as unrepeatable read. When a transaction calculates some summary function over a set of data while the other transactions are updating the data, then the Inconsistent Retrievals Problem occurs.
* A transaction T1 reads a record and then does some other processing during which the transaction T2 updates the record. Now when the transaction T1 reads the record, then the new value will be inconsistent with the previous value.

**Example:**

Suppose two transactions operate on three accounts.

  
  
Transaction-X is doing the sum of all balance while transaction-Y is transferring an amount 50 from Account-1 to Account-3.

* Here, transaction-X produces the result of 550 which is incorrect. If we write this produced result in the database, the database will become an inconsistent state because the actual sum is 600.
* Here, transaction-X has seen an inconsistent state of the database.

## Concurrency Control Protocol

Concurrency control protocols ensure atomicity, isolation, and serializability of concurrent transactions. The concurrency control protocol can be divided into three categories:

1. Lock based protocol
2. Time-stamp protocol
3. Validation based protocol

# **Lock-Based Protocol**

In this type of protocol, any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock:

**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.

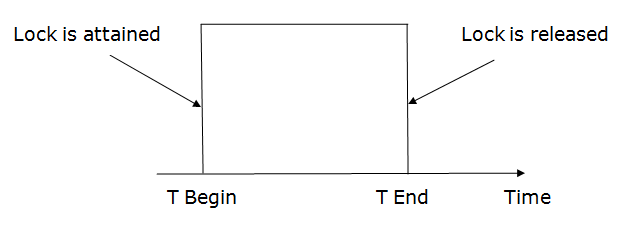
## There are four types of lock protocols available:

### **1. Simplistic lock protocol**

It is the simplest way of locking the data while transaction. Simplistic lock-based protocols allow all the transactions to get the lock on the data before insert or delete or update on it. It will unlock the data item after completing the transaction.

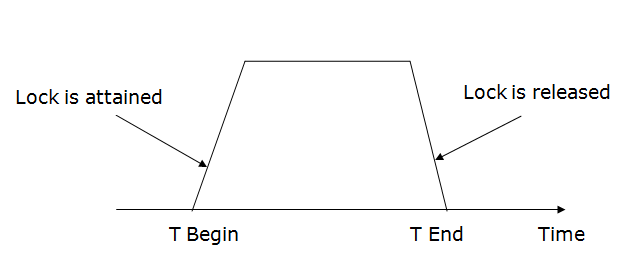
### **2. Pre-claiming Lock Protocol**

* Pre-claiming Lock Protocols evaluate the transaction to list all the data items on which they need locks.
* Before initiating an execution of the transaction, it requests DBMS for all the lock on all those data items.
* If all the locks are granted then this protocol allows the transaction to begin. When the transaction is completed then it releases all the lock.
* If all the locks are not granted then this protocol allows the transaction to rolls back and waits until all the locks are granted.



### **3. Two-phase locking (2PL)**

* The two-phase locking protocol divides the execution phase of the transaction into three parts.
* In the first part, when the execution of the transaction starts, it seeks permission for the lock it requires.
* In the second part, the transaction acquires all the locks. The third phase is started as soon as the transaction releases its first lock.
* In the third phase, the transaction cannot demand any new locks. It only releases the acquired locks.



There are two phases of 2PL:

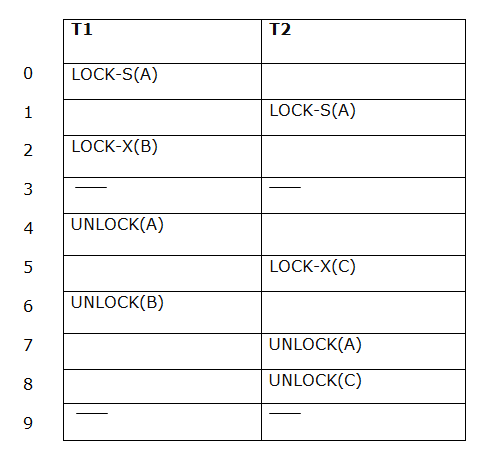
**Growing phase:** In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

**Shrinking phase:** In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

In the below example, if lock conversion is allowed then the following phase can happen:

1. Upgrading of lock (from S(a) to X (a)) is allowed in growing phase.
2. Downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.

**Example:**

The following way shows how unlocking and locking work with 2-PL.

**Transaction T1:**

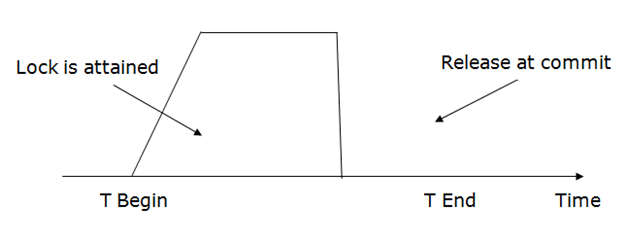
* **Growing phase:** from step 1-3
* **Shrinking phase:** from step 5-7
* **Lock point:** at 3

**Transaction T2:**

* **Growing phase:** from step 2-6
* **Shrinking phase:** from step 8-9
* **Lock point:** at 6

### **4. Strict Two-phase locking (Strict-2PL)**

* The first phase of Strict-2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally.
* The only difference between 2PL and strict 2PL is that Strict-2PL does not release a lock after using it.
* Strict-2PL waits until the whole transaction to commit, and then it releases all the locks at a time.
* Strict-2PL protocol does not have shrinking phase of lock release.



It does not have cascading abort as 2PL does.

# **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.

**Basic Timestamp ordering protocol works as follows:**

1. Check the following condition whenever a transaction Ti issues a **Read (X)** operation:

* If W\_TS(X) >TS(Ti) then the operation is rejected.
* If W\_TS(X) <= TS(Ti) then the operation is executed.
* Timestamps of all the data items are updated.

2. Check the following condition whenever a transaction Ti issues a **Write(X)** operation:

* If TS(Ti) < R\_TS(X) then the operation is rejected.
* If TS(Ti) < W\_TS(X) then the operation is rejected and Ti is rolled back otherwise the operation is executed.

**Where,**

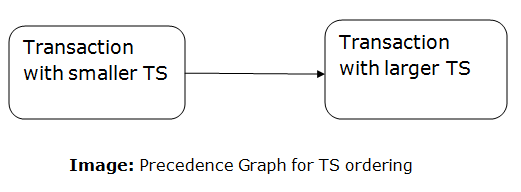
**TS(TI)** denotes the timestamp of the transaction Ti.

**R\_TS(X)** denotes the Read time-stamp of data-item X.

**W\_TS(X)** denotes the Write time-stamp of data-item X.

## Advantages and Disadvantages of TO protocol:

* TO protocol ensures serializability since the precedence graph is as follows:

* TS protocol ensures freedom from deadlock that means no transaction ever waits.
* But the schedule may not be recoverable and may not even be cascade- free.

Validation Based Protocol

Validation phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases:

1. **Read phase:** In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.
2. **Validation phase:** In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.
3. **Write phase:** If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Here each phase has the following different timestamps:

**Start(Ti):** It contains the time when Ti started its execution.

**Validation (Ti):** It contains the time when Ti finishes its read phase and starts its validation phase.

**Finish(Ti):** It contains the time when Ti finishes its write phase.

* This protocol is used to determine the time stamp for the transaction for serialization using the time stamp of the validation phase, as it is the actual phase which determines if the transaction will commit or rollback.
* Hence TS(T) = validation(T).
* The serializability is determined during the validation process. It can't be decided in advance.
* While executing the transaction, it ensures a greater degree of concurrency and also less number of conflicts.
* Thus it contains transactions which have less number of rollbacks.

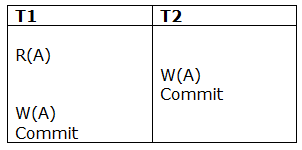
Thomas write Rule

Thomas Write Rule provides the guarantee of serializability order for the protocol. It improves the Basic Timestamp Ordering Algorithm.

The basic Thomas write rules are as follows:

* If TS(T) < R\_TS(X) then transaction T is aborted and rolled back, and operation is rejected.
* If TS(T) < W\_TS(X) then don't execute the W\_item(X) operation of the transaction and continue processing.
* If neither condition 1 nor condition 2 occurs, then allowed to execute the WRITE operation by transaction Ti and set W\_TS(X) to TS(T).

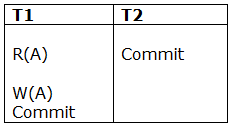
If we use the Thomas write rule then some serializable schedule can be permitted that does not conflict serializable as illustrate by the schedule in a given figure:



**Figure:** A Serializable Schedule that is not Conflict Serializable

In the above figure, T1's read and precedes T1's write of the same data item. This schedule does not conflict serializable.

Thomas write rule checks that T2's write is never seen by any transaction. If we delete the write operation in transaction T2, then conflict serializable schedule can be obtained which is shown in below figure.

  
**Figure:** A Conflict Serializable Schedule

# **Multiple Granularity**

Let's start by understanding the meaning of granularity.

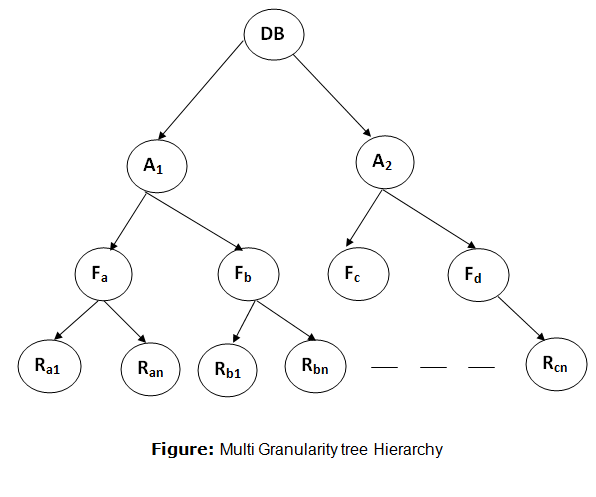
**Granularity:** It is the size of data item allowed to lock.

### **Multiple Granularity:**

* It can be defined as hierarchically breaking up the database into blocks which can be locked.
* The Multiple Granularity protocol enhances concurrency and reduces lock overhead.
* It maintains the track of what to lock and how to lock.
* It makes easy to decide either to lock a data item or to unlock a data item. This type of hierarchy can be graphically represented as a tree.

**For example:** Consider a tree which has four levels of nodes.

* The first level or higher level shows the entire database.
* The second level represents a node of type area. The higher level database consists of exactly these areas.
* The area consists of children nodes which are known as files. No file can be present in more than one area.
* Finally, each file contains child nodes known as records. The file has exactly those records that are its child nodes. No records represent in more than one file.
* Hence, the levels of the tree starting from the top level are as follows:
  1. Database
  2. Area
  3. File
  4. Record

  
 In this example, the highest level shows the entire database. The levels below are file, record, and fields.

There are three additional lock modes with multiple granularity:

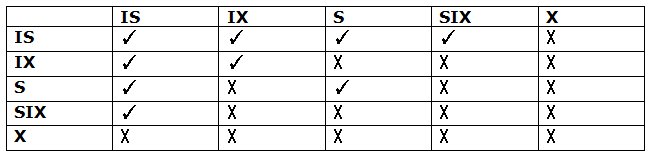
## Intention Mode Lock

**Intention-shared (IS):** It contains explicit locking at a lower level of the tree but only with shared locks.

**Intention-Exclusive (IX):** It contains explicit locking at a lower level with exclusive or shared locks.

**Shared & Intention-Exclusive (SIX):** In this lock, the node is locked in shared mode, and some node is locked in exclusive mode by the same transaction.

**Compatibility Matrix with Intention Lock Modes:** The below table describes the compatibility matrix for these lock modes:



It uses the intention lock modes to ensure serializability. It requires that if a transaction attempts to lock a node, then that node must follow these protocols:

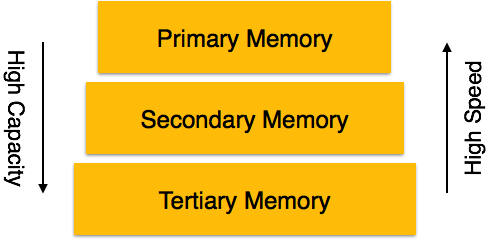
* Transaction T1 should follow the lock-compatibility matrix.
* Transaction T1 firstly locks the root of the tree. It can lock it in any mode.
* If T1 currently has the parent of the node locked in either IX or IS mode, then the transaction T1 will lock a node in S or IS mode only.
* If T1 currently has the parent of the node locked in either IX or SIX modes, then the transaction T1 will lock a node in X, SIX, or IX mode only.
* If T1 has not previously unlocked any node only, then the Transaction T1 can lock a node.
* If T1 currently has none of the children of the node-locked only, then Transaction T1 will unlock a node.

Observe that in multiple-granularity, the locks are acquired in top-down order, and locks must be released in bottom-up order.

* If transaction T1 reads record Ra9 in file Fa, then transaction T1 needs to lock the database, area A1 and file Fa in IX mode. Finally, it needs to lock Ra2 in S mode.
* If transaction T2 modifies record Ra9 in file Fa, then it can do so after locking the database, area A1 and file Fa in IX mode. Finally, it needs to lock the Ra9 in X mode.
* If transaction T3 reads all the records in file Fa, then transaction T3 needs to lock the database, and area A in IS mode. At last, it needs to lock Fa in S mode.
* If transaction T4 reads the entire database, then T4 needs to lock the database in S mode.

UNIT-V

Databases are stored in file formats, which contain records. At physical level, the actual data is stored in electromagnetic format on some device. These storage devices can be broadly categorized into three types −



* **Primary Storage** − The memory storage that is directly accessible to the CPU comes under this category. CPU's internal memory (registers), fast memory (cache), and main memory (RAM) are directly accessible to the CPU, as they are all placed on the motherboard or CPU chipset. This storage is typically very small, ultra-fast, and volatile. Primary storage requires continuous power supply in order to maintain its state. In case of a power failure, all its data is lost.
* **Secondary Storage** − Secondary storage devices are used to store data for future use or as backup. Secondary storage includes memory devices that are not a part of the CPU chipset or motherboard, for example, magnetic disks, optical disks (DVD, CD, etc.), hard disks, flash drives, and magnetic tapes.
* **Tertiary Storage** − Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are the slowest in speed. These storage devices are mostly used to take the back up of an entire system. Optical disks and magnetic tapes are widely used as tertiary storage.

## Memory Hierarchy

A computer system has a well-defined hierarchy of memory. A CPU has direct access to it main memory as well as its inbuilt registers. The access time of the main memory is obviously less than the CPU speed. To minimize this speed mismatch, cache memory is introduced. Cache memory provides the fastest access time and it contains data that is most frequently accessed by the CPU.

The memory with the fastest access is the costliest one. Larger storage devices offer slow speed and they are less expensive, however they can store huge volumes of data as compared to CPU registers or cache memory.

## Magnetic Disks

Hard disk drives are the most common secondary storage devices in present computer systems. These are called magnetic disks because they use the concept of magnetization to store information. Hard disks consist of metal disks coated with magnetizable material. These disks are placed vertically on a spindle. A read/write head moves in between the disks and is used to magnetize or de-magnetize the spot under it. A magnetized spot can be recognized as 0 (zero) or 1 (one).

Hard disks are formatted in a well-defined order to store data efficiently. A hard disk plate has many concentric circles on it, called **tracks**. Every track is further divided into **sectors**. A sector on a hard disk typically stores 512 bytes of data.

# **RAID**

RAID refers to redundancy array of the independent disk. It is a technology which is used to connect multiple secondary storage devices for increased performance, data redundancy or both. It gives you the ability to survive one or more drive failure depending upon the RAID level used.

It consists of an array of disks in which multiple disks are connected to achieve different goals.

## RAID technology

There are 7 levels of RAID schemes. These schemas are as RAID 0, RAID 1, ...., RAID 6.

These levels contain the following characteristics:

* It contains a set of physical disk drives.
* In this technology, the operating system views these separate disks as a single logical disk.
* In this technology, data is distributed across the physical drives of the array.
* Redundancy disk capacity is used to store parity information.
* In case of disk failure, the parity information can be helped to recover the data.

## Standard RAID levels

### **RAID 0**

* RAID level 0 provides data stripping, i.e., a data can place across multiple disks. It is based on stripping that means if one disk fails then all data in the array is lost.
* This level doesn't provide fault tolerance but increases the system performance.

## Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **Disk 0** | **Disk 1** | **Disk 2** | **Disk 3** |
| 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 |

In this figure, block 0, 1, 2, 3 form a stripe.

In this level, instead of placing just one block into a disk at a time, we can work with two or more blocks placed it into a disk before moving on to the next one.

|  |  |  |  |
| --- | --- | --- | --- |
| **Disk 0** | **Disk 1** | **Disk 2** | **Disk 3** |
| 20 | 22 | 24 | 26 |
| 21 | 23 | 25 | 27 |
| 28 | 30 | 32 | 34 |
| 29 | 31 | 33 | 35 |

In this above figure, there is no duplication of data. Hence, a block once lost cannot be recovered.

### **Pros of RAID 0:**

* In this level, throughput is increased because multiple data requests probably not on the same disk.
* This level full utilizes the disk space and provides high performance.
* It requires minimum 2 drives.

### **Cons of RAID 0:**

* It doesn't contain any error detection mechanism.
* The RAID 0 is not a true RAID because it is not fault-tolerance.
* In this level, failure of either disk results in complete data loss in respective array.

## RAID 1

This level is called mirroring of data as it copies the data from drive 1 to drive 2. It provides 100% redundancy in case of a failure.

## Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **Disk 0** | **Disk 1** | **Disk 2** | **Disk 3** |
| A | A | B | B |
| C | C | D | D |
| E | E | F | F |
| G | G | H | H |

Only half space of the drive is used to store the data. The other half of drive is just a mirror to the already stored data.

### **Pros of RAID 1:**

* The main advantage of RAID 1 is fault tolerance. In this level, if one disk fails, then the other automatically takes over.
* In this level, the array will function even if any one of the drives fails.

### **Cons of RAID 1:**

* In this level, one extra drive is required per drive for mirroring, so the expense is higher.

## RAID 2

* RAID 2 consists of bit-level striping using hamming code parity. In this level, each data bit in a word is recorded on a separate disk and ECC code of data words is stored on different set disks.
* Due to its high cost and complex structure, this level is not commercially used. This same performance can be achieved by RAID 3 at a lower cost.

### **Pros of RAID 2:**

* This level uses one designated drive to store parity.
* It uses the hamming code for error detection.

### **Cons of RAID 2:**

* It requires an additional drive for error detection.

## RAID 3

* RAID 3 consists of byte-level striping with dedicated parity. In this level, the parity information is stored for each disk section and written to a dedicated parity drive.
* In case of drive failure, the parity drive is accessed, and data is reconstructed from the remaining devices. Once the failed drive is replaced, the missing data can be restored on the new drive.
* In this level, data can be transferred in bulk. Thus high-speed data transmission is possible.

|  |  |  |  |
| --- | --- | --- | --- |
| **Disk 0** | **Disk 1** | **Disk 2** | **Disk 3** |
| A | B | C | P(A, B, C) |
| D | E | F | P(D, E, F) |
| G | H | I | P(G, H, I) |
| J | K | L | P(J, K, L) |

### **Pros of RAID 3:**

* In this level, data is regenerated using parity drive.
* It contains high data transfer rates.
* In this level, data is accessed in parallel.

### **Cons of RAID 3:**

* It required an additional drive for parity.
* It gives a slow performance for operating on small sized files.

## RAID 4

* RAID 4 consists of block-level stripping with a parity disk. Instead of duplicating data, the RAID 4 adopts a parity-based approach.
* This level allows recovery of at most 1 disk failure due to the way parity works. In this level, if more than one disk fails, then there is no way to recover the data.
* Level 3 and level 4 both are required at least three disks to implement RAID.

|  |  |  |  |
| --- | --- | --- | --- |
| **Disk 0** | **Disk 1** | **Disk 2** | **Disk 3** |
| A | B | C | P0 |
| D | E | F | P1 |
| G | H | I | P2 |
| J | K | L | P3 |

In this figure, we can observe one disk dedicated to parity.

In this level, parity can be calculated using an XOR function. If the data bits are 0,0,0,1 then the parity bits is XOR(0,1,0,0) = 1. If the parity bits are 0,0,1,1 then the parity bit is XOR(0,0,1,1)= 0. That means, even number of one results in parity 0 and an odd number of one results in parity 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C1** | **C2** | **C3** | **C4** | **Parity** |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |

Suppose that in the above figure, C2 is lost due to some disk failure. Then using the values of all the other columns and the parity bit, we can recompute the data bit stored in C2. This level allows us to recover lost data.

## RAID 5

* RAID 5 is a slight modification of the RAID 4 system. The only difference is that in RAID 5, the parity rotates among the drives.
* It consists of block-level striping with DISTRIBUTED parity.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Disk 0** | **Disk 1** | **Disk 2** | **Disk 3** | **Disk 4** |
| 0 | 1 | 2 | 3 | P0 |
| 5 | 6 | 7 | P1 | 4 |
| 10 | 11 | P2 | 8 | 9 |
| 15 | P3 | 12 | 13 | 14 |
| P4 | 16 | 17 | 18 | 19 |

* Same as RAID 4, this level allows recovery of at most 1 disk failure. If more than one disk fails, then there is no way for data recovery.

This figure shows that how parity bit rotates.

This level was introduced to make the random write performance better.

### **Pros of RAID 5:**

* This level is cost effective and provides high performance.
* In this level, parity is distributed across the disks in an array.
* It is used to make the random write performance better.

### **Cons of RAID 5:**

* In this level, disk failure recovery takes longer time as parity has to be calculated from all available drives.
* This level cannot survive in concurrent drive failure.

## RAID 6

* This level is an extension of RAID 5. It contains block-level stripping with 2 parity bits.
* In RAID 6, you can survive 2 concurrent disk failures. Suppose you are using RAID 5, and RAID 1. When your disks fail, you need to replace the failed disk because if simultaneously another disk fails then you won't be able to recover any of the data, so in this case RAID 6 plays its part where you can survive two concurrent disk failures before you run out of options.

|  |  |  |  |
| --- | --- | --- | --- |
| **Disk 1** | **Disk 2** | **Disk 3** | **Disk 4** |
| A0 | B0 | Q0 | P0 |
| A1 | Q1 | P1 | D1 |
| Q2 | P2 | C2 | D2 |
| P3 | B3 | C3 | Q3 |

### **Pros of RAID 6:**

* This level performs RAID 0 to strip data and RAID 1 to mirror. In this level, stripping is performed before mirroring.
* In this level, drives required should be multiple of 2.

### **Cons of RAID 6:**

* It is not utilized 100% disk capability as half is used for mirroring.
* It contains very limited scalability.

# **File Organization**

* The **File** is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.
* File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
* File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
* The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.
* Files of fixed length records are easier to implement than the files of variable length records.

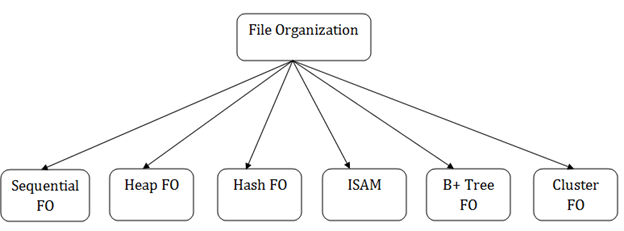
## Objective of file organization

* It contains an optimal selection of records, i.e., records can be selected as fast as possible.
* To perform insert, delete or update transaction on the records should be quick and easy.
* The duplicate records cannot be induced as a result of insert, update or delete.
* For the minimal cost of storage, records should be stored efficiently.

## Types of file organization:

File organization contains various methods. These particular methods have pros and cons on the basis of access or selection. In the file organization, the programmer decides the best-suited file organization method according to his requirement.

Types of file organization are as follows:

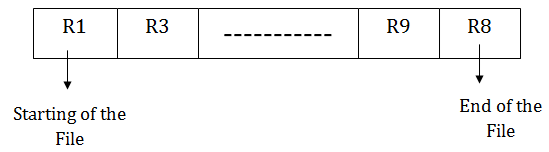


# **Sequential File Organization**

This method is the easiest method for file organization. In this method, files are stored sequentially. This method can be implemented in two ways:

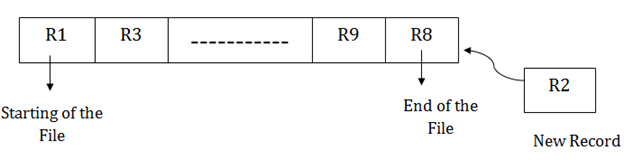
## 1. Pile File Method:

* It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables.
* In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.



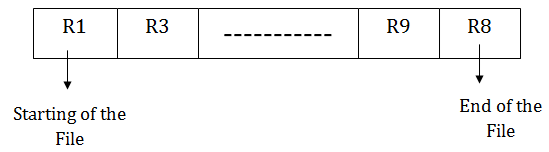
## Insertion of the new record:

Suppose we have four records R1, R3 and so on upto R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2 in the sequence, then it will be placed at the end of the file. Here, records are nothing but a row in any table.



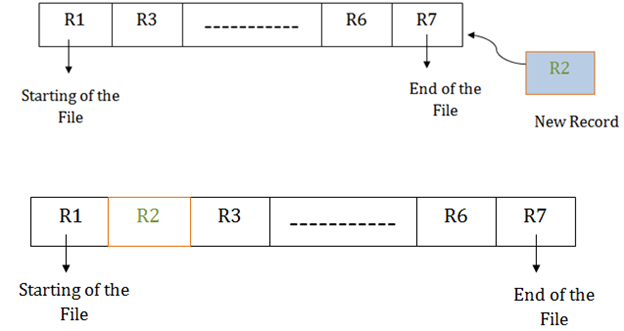
## 2. Sorted File Method:

* In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order. Sorting of records is based on any primary key or any other key.
* In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.



## Insertion of the new record:

Suppose there is a preexisting sorted sequence of four records R1, R3 and so on upto R6 and R7. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence.



## Pros of sequential file organization

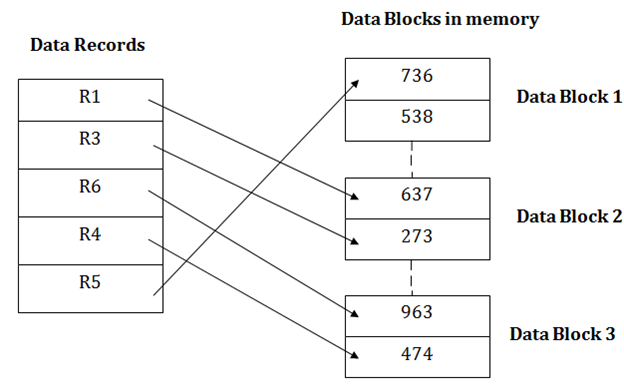
* It contains a fast and efficient method for the huge amount of data.
* In this method, files can be easily stored in cheaper storage mechanism like magnetic tapes.
* It is simple in design. It requires no much effort to store the data.
* This method is used when most of the records have to be accessed like grade calculation of a student, generating the salary slip, etc.
* This method is used for report generation or statistical calculations.

## Cons of sequential file organization

* It will waste time as we cannot jump on a particular record that is required but we have to move sequentially which takes our time.
* Sorted file method takes more time and space for sorting the records.

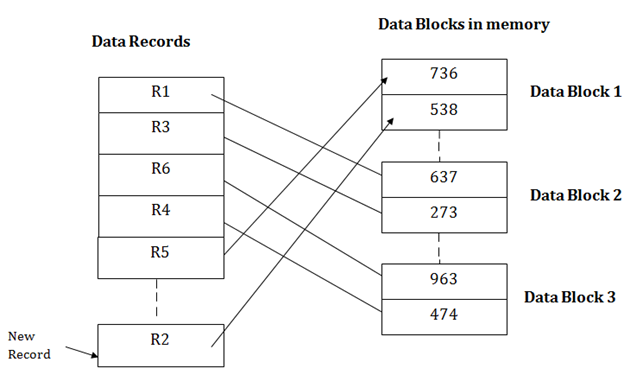
# **Heap file organization**

* It is the simplest and most basic type of organization. It works with data blocks. In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
* When the data block is full, the new record is stored in some other block. This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file.
* In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.



## Insertion of a new record

Suppose we have five records R1, R3, R6, R4 and R5 in a heap and suppose we want to insert a new record R2 in a heap. If the data block 3 is full then it will be inserted in any of the database selected by the DBMS, let's say data block 1.



If we want to search, update or delete the data in heap file organization, then we need to traverse the data from staring of the file till we get the requested record.

If the database is very large then searching, updating or deleting of record will be time-consuming because there is no sorting or ordering of records. In the heap file organization, we need to check all the data until we get the requested record.

## Pros of Heap file organization

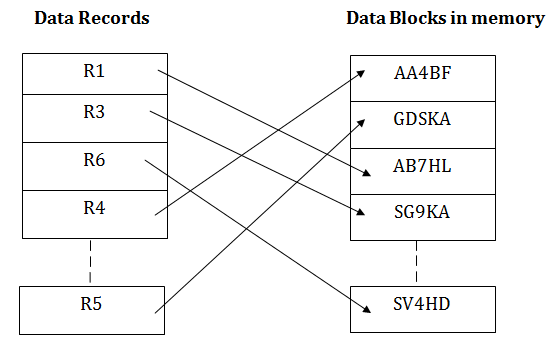
* It is a very good method of file organization for bulk insertion. If there is a large number of data which needs to load into the database at a time, then this method is best suited.
* In case of a small database, fetching and retrieving of records is faster than the sequential record.

## Cons of Heap file organization

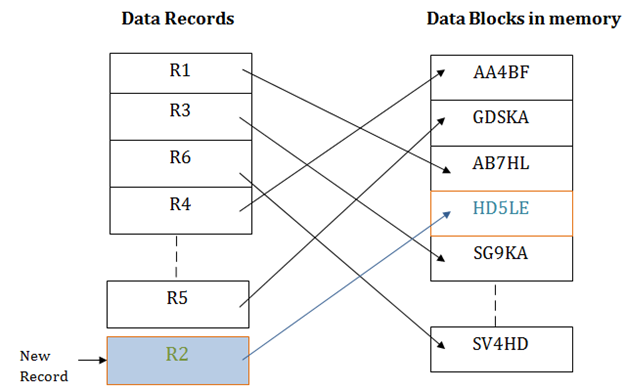
* This method is inefficient for the large database because it takes time to search or modify the record.
* This method is inefficient for large databases.

Hash File Organization

* Hash File Organization uses the computation of hash function on some fields of the records. The hash function's output determines the location of disk block where the records are to be placed.

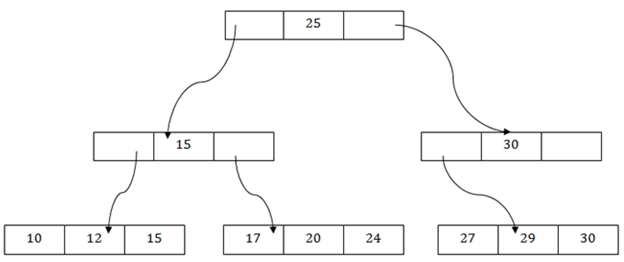


* When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address. In the same way, when a new record has to be inserted, then the address is generated using the hash key and record is directly inserted. The same process is applied in the case of delete and update.
* In this method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.



# **B+ File Organization**

* B+ tree file organization is the advanced method of an indexed sequential access method. It uses a tree-like structure to store records in File.
* It uses the same concept of key-index where the primary key is used to sort the records. For each primary key, the value of the index is generated and mapped with the record.
* The B+ tree is similar to a binary search tree (BST), but it can have more than two children. In this method, all the records are stored only at the leaf node. Intermediate nodes act as a pointer to the leaf nodes. They do not contain any records.



## The above B+ tree shows that:

* There is one root node of the tree, i.e., 25.
* There is an intermediary layer with nodes. They do not store the actual record. They have only pointers to the leaf node.
* The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root, i.e., 15 and 30 respectively.
* There is only one leaf node which has only values, i.e., 10, 12, 17, 20, 24, 27 and 29.
* Searching for any record is easier as all the leaf nodes are balanced.
* In this method, searching any record can be traversed through the single path and accessed easily.

## Adv of B+ tree file organization

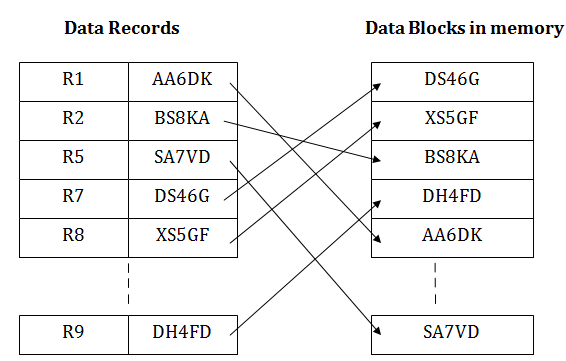
* In this method, searching becomes very easy as all the records are stored only in the leaf nodes and sorted the sequential linked list.
* Traversing through the tree structure is easier and faster.
* The size of the B+ tree has no restrictions, so the number of records can increase or decrease and the B+ tree structure can also grow or shrink.
* It is a balanced tree structure, and any insert/update/delete does not affect the performance of tree.

## Disadv of B+ tree file organization

* This method is inefficient for the static method.

# **Indexed sequential access method (ISAM)**

ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key. An index value is generated for each primary key and mapped with the record. This index contains the address of the record in the file.



If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

## Adv of ISAM:

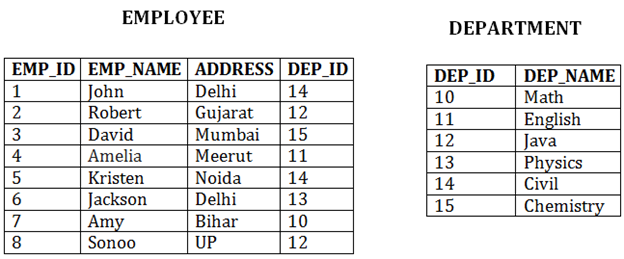
* In this method, each record has the address of its data block, searching a record in a huge database is quick and easy.
* This method supports range retrieval and partial retrieval of records. Since the index is based on the primary key values, we can retrieve the data for the given range of value. In the same way, the partial value can also be easily searched, i.e., the student name starting with 'JA' can be easily searched.

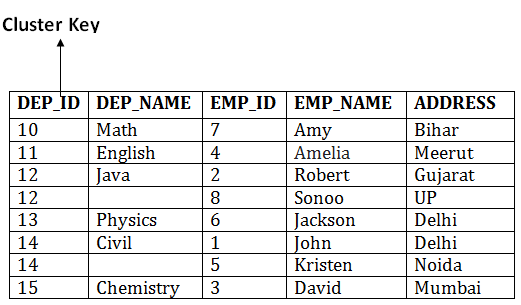
## Disadv of ISAM

* This method requires extra space in the disk to store the index value.
* When the new records are inserted, then these files have to be reconstructed to maintain the sequence.
* When the record is deleted, then the space used by it needs to be released. Otherwise, the performance of the database will slow down.

# **Cluster file organization**

* When the two or more records are stored in the same file, it is known as clusters. These files will have two or more tables in the same data block, and key attributes which are used to map these tables together are stored only once.
* This method reduces the cost of searching for various records in different files.
* The cluster file organization is used when there is a frequent need for joining the tables with the same condition. These joins will give only a few records from both tables. In the given example, we are retrieving the record for only particular departments. This method can't be used to retrieve the record for the entire department.





In this method, we can directly insert, update or delete any record. Data is sorted based on the key with which searching is done. Cluster key is a type of key with which joining of the table is performed.

**Cluster file organization is of two types:**

## 1. Indexed Clusters:

In indexed cluster, records are grouped based on the cluster key and stored together. The above EMPLOYEE and DEPARTMENT relationship is an example of an indexed cluster. Here, all the records are grouped based on the cluster key- DEP\_ID and all the records are grouped.

## 2. Hash Clusters:

It is similar to the indexed cluster. In hash cluster, instead of storing the records based on the cluster key, we generate the value of the hash key for the cluster key and store the records with the same hash key value.

## Pros of Cluster file organization

* The cluster file organization is used when there is a frequent request for joining the tables with same joining condition.
* It provides the efficient result when there is a 1:M mapping between the tables.

## Cons of Cluster file organization

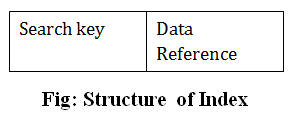
* This method has the low performance for the very large database.
* If there is any change in joining condition, then this method cannot use. If we change the condition of joining then traversing the file takes a lot of time.
* This method is not suitable for a table with a 1:1 condition.

# **Indexing:**

* 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.

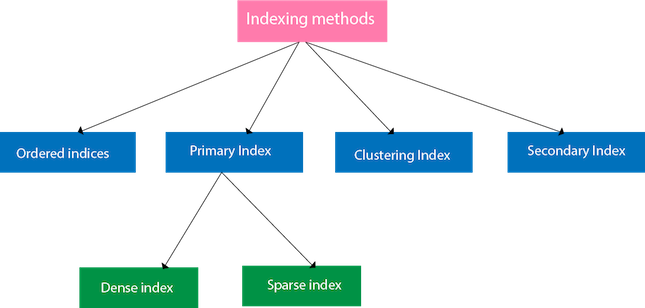
## Index structure:

Indexes can be created using some database columns.



* The first column of the database is the search key that contains a copy of the primary key or candidate key of the table. The values of the primary key are stored in sorted order so that the corresponding data can be accessed easily.
* The second column of the database is the data reference. It contains a set of pointers holding the address of the disk block where the value of the particular key can be found.

## Indexing Methods



## Ordered indices

The indices are usually sorted to make searching faster. The indices which are sorted are known as ordered indices.

**Example**: Suppose we have an employee table with thousands of record and each of which is 10 bytes long. If their IDs start with 1, 2, 3....and so on and we have to search student with ID-543.

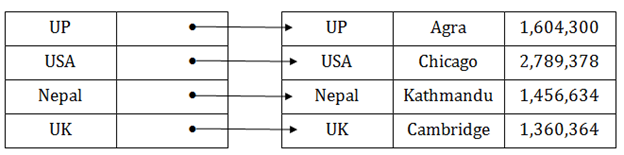
* In the case of a database with no index, we have to search the disk block from starting till it reaches 543. The DBMS will read the record after reading 543\*10=5430 bytes.
* In the case of an index, we will search using indexes and the DBMS will read the record after reading 542\*2= 1084 bytes which are very less compared to the previous case.

## Primary Index

* If the index is created on the basis of the primary key of the table, then it is known as primary indexing. These primary keys are unique to each record and contain 1:1 relation between the records.
* As primary keys are stored in sorted order, the performance of the searching operation is quite efficient.
* The primary index can be classified into two types: Dense index and Sparse index.

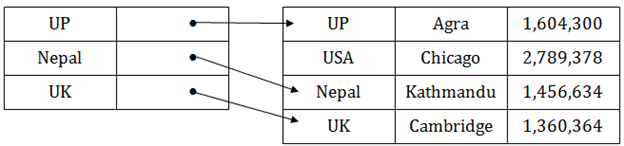
## Dense index

* The dense index contains an index record for every search key value in the data file. It makes searching faster.
* In this, the number of records in the index table is same as the number of records in the main table.
* It needs more space to store index record itself. The index records have the search key and a pointer to the actual record on the disk.



## Sparse index

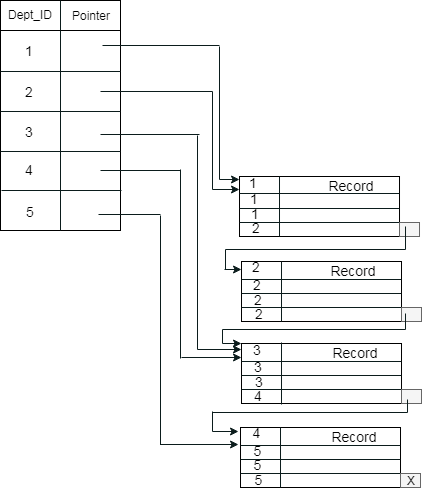
* In the data file, index record appears only for a few items. Each item points to a block.
* In this, instead of pointing to each record in the main table, the index points to the records in the main table in a gap.



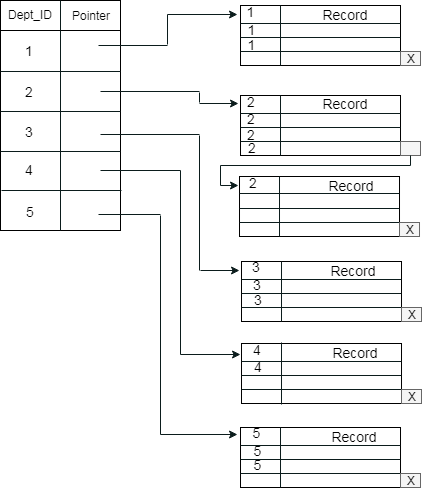
## Clustering Index

* A clustered index can be defined as an ordered data file. Sometimes the index is created on non-primary key columns which may not be unique for each record.
* In this case, to identify the record faster, we will group two or more columns to get the unique value and create index out of them. This method is called a clustering index.
* The records which have similar characteristics are grouped, and indexes are created for these group.

**Example**: suppose a company contains several employees in each department. Suppose we use a clustering index, where all employees which belong to the same Dept\_ID are considered within a single cluster, and index pointers point to the cluster as a whole. Here Dept\_Id is a non-unique key.



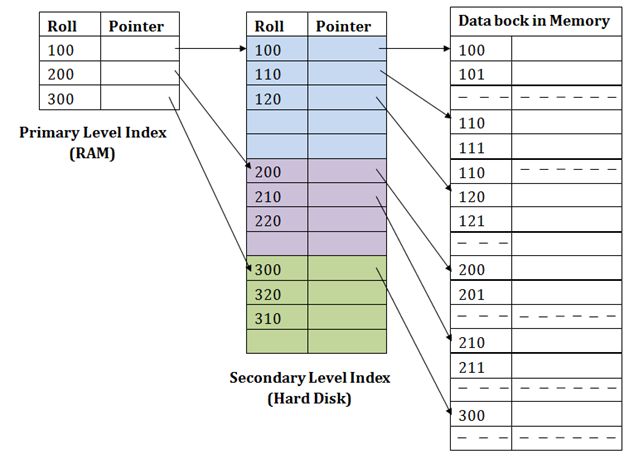
The previous schema is little confusing because one disk block is shared by records which belong to the different cluster. If we use separate disk block for separate clusters, then it is called better technique.



## Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk).



**For example:**

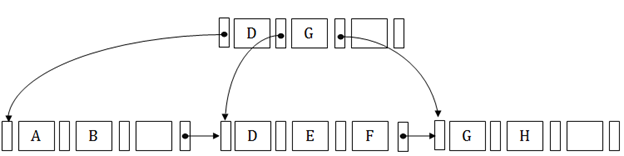
* If you want to find the record of roll 111 in the diagram, then it will search the highest entry which is smaller than or equal to 111 in the first level index. It will get 100 at this level.
* Then in the second index level, again it does max (111) <= 111 and gets 110. Now using the address 110, it goes to the data block and starts searching each record till it gets 111.
* This is how a search is performed in this method. Inserting, updating or deleting is also done in the same manner.

# **B+ Tree**

* The B+ tree is a balanced binary search tree. It follows a multi-level index format.
* In the B+ tree, leaf nodes denote actual data pointers. B+ tree ensures that all leaf nodes remain at the same height.
* In the B+ tree, the leaf nodes are linked using a link list. Therefore, a B+ tree can support random access as well as sequential access.

## Structure of B+ Tree

* In the B+ tree, every leaf node is at equal distance from the root node. The B+ tree is of the order n where n is fixed for every B+ tree.
* It contains an internal node and leaf node.



## Internal node

* An internal node of the B+ tree can contain at least n/2 record pointers except the root node.
* At most, an internal node of the tree contains n pointers.

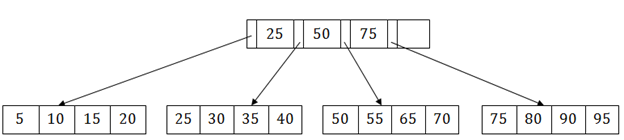
## Leaf node

* The leaf node of the B+ tree can contain at least n/2 record pointers and n/2 key values.
* At most, a leaf node contains n record pointer and n key values.
* Every leaf node of the B+ tree contains one block pointer P to point to next leaf node.

## Searching a record in B+ Tree

Suppose we have to search 55 in the below B+ tree structure. First, we will fetch for the intermediary node which will direct to the leaf node that can contain a record for 55.

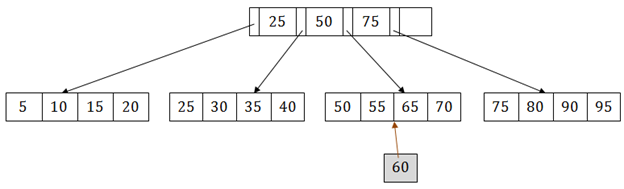
So, in the intermediary node, we will find a branch between 50 and 75 nodes. Then at the end, we will be redirected to the third leaf node. Here DBMS will perform a sequential search to find 55.



## B+ Tree Insertion

Suppose we want to insert a record 60 in the below structure. It will go to the 3rd leaf node after 55. It is a balanced tree, and a leaf node of this tree is already full, so we cannot insert 60 there.

In this case, we have to split the leaf node, so that it can be inserted into tree without affecting the fill factor, balance and order.



The 3rd leaf node has the values (50, 55, 60, 65, 70) and its current root node is 50. We will split the leaf node of the tree in the middle so that its balance is not altered. So we can group (50, 55) and (60, 65, 70) into 2 leaf nodes.

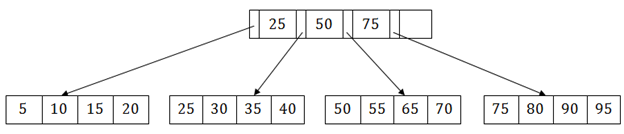
If these two has to be leaf nodes, the intermediate node cannot branch from 50. It should have 60 added to it, and then we can have pointers to a new leaf node.

This is how we can insert an entry when there is overflow. In a normal scenario, it is very easy to find the node where it fits and then place it in that leaf node.

## B+ Tree Deletion

Suppose we want to delete 60 from the above example. In this case, we have to remove 60 from the intermediate node as well as from the 4th leaf node too. If we remove it from the intermediate node, then the tree will not satisfy the rule of the B+ tree. So we need to modify it to have a balanced tree.

After deleting node 60 from above B+ tree and re-arranging the nodes, it will show as follows:

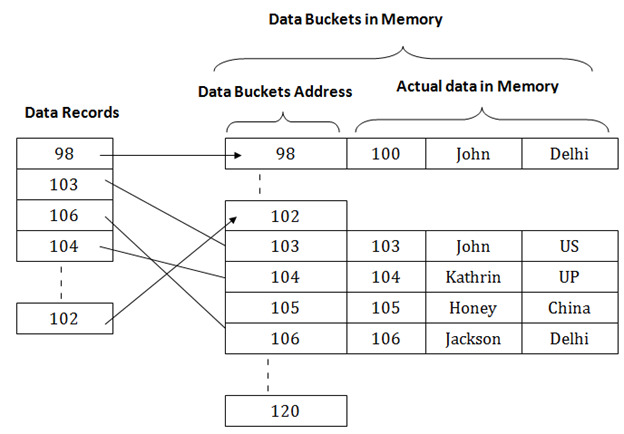


# **Hashing**

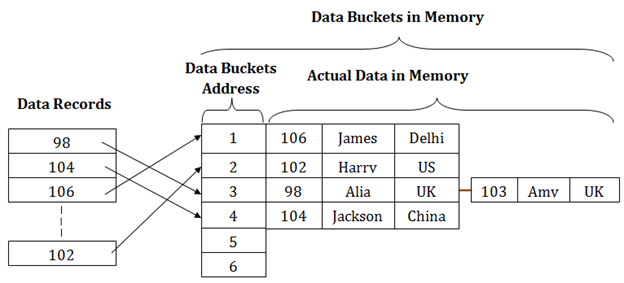
In a huge database structure, it is very inefficient to search all the index values and reach the desired data. Hashing technique is used to calculate the direct location of a data record on the disk without using index structure.

In this technique, data is stored at the data blocks whose address is generated by using the hashing function. The memory location where these records are stored is known as data bucket or data blocks.

In this, a hash function can choose any of the column value to generate the address. Most of the time, the hash function uses the primary key to generate the address of the data block. A hash function is a simple mathematical function to any complex mathematical function. We can even consider the primary key itself as the address of the data block. That means each row whose address will be the same as a primary key stored in the data block.



The above diagram shows data block addresses same as primary key value. This hash function can also be a simple mathematical function like exponential, mod, cos, sin, etc. Suppose we have mod (5) hash function to determine the address of the data block. In this case, it applies mod (5) hash function on the primary keys and generates 3, 3, 1, 4 and 2 respectively, and records are stored in those data block addresses.



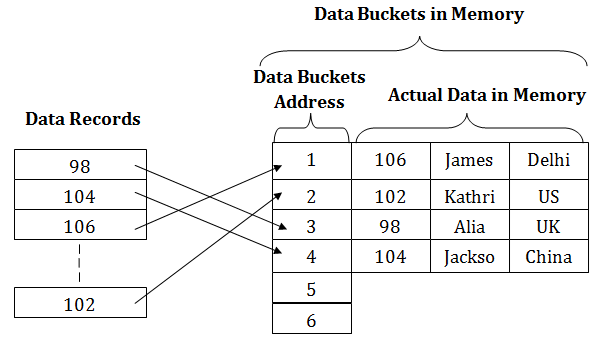
## Types of Hashing:

* [Static Hashing](https://www.javatpoint.com/dbms-static-hashing)
* [Dynamic Hashing](https://www.javatpoint.com/dbms-dynamic-hashing)

# **Static Hashing**

In static hashing, the resultant data bucket address will always be the same. That means if we generate an address for EMP\_ID =103 using the hash function mod (5) then it will always result in same bucket address 3. Here, there will be no change in the bucket address.

Hence in this static hashing, the number of data buckets in memory remains constant throughout. In this example, we will have five data buckets in the memory used to store the data.



## Operations of Static Hashing

* **Searching a record**

When a record needs to be searched, then the same hash function retrieves the address of the bucket where the data is stored.

* **Insert a Record**

When a new record is inserted into the table, then we will generate an address for a new record based on the hash key and record is stored in that location.

* **Delete a Record**

To delete a record, we will first fetch the record which is supposed to be deleted. Then we will delete the records for that address in memory.

* **Update a Record**

To update a record, we will first search it using a hash function, and then the data record is updated.

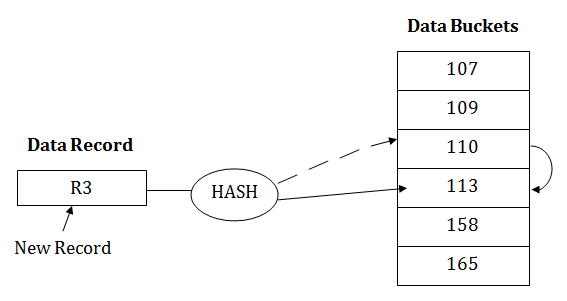
If we want to insert some new record into the file but the address of a data bucket generated by the hash function is not empty, or data already exists in that address. This situation in the static hashing is known as **bucket overflow**. This is a critical situation in this method.

To overcome this situation, there are various methods. Some commonly used methods are as follows:

## 1. Open Hashing

When a hash function generates an address at which data is already stored, then the next bucket will be allocated to it. This mechanism is called as **Linear Probing**.

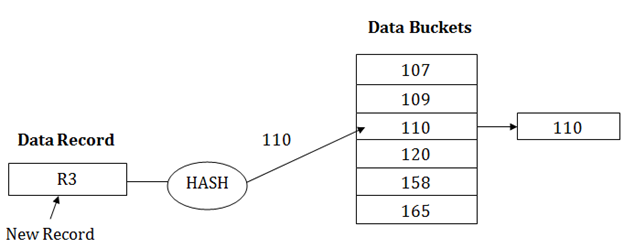
**For example:** suppose R3 is a new address which needs to be inserted, the hash function generates address as 112 for R3. But the generated address is already full. So the system searches next available data bucket, 113 and assigns R3 to it.



## 2. Close Hashing

When buckets are full, then a new data bucket is allocated for the same hash result and is linked after the previous one. This mechanism is known as **Overflow chaining**.

**For example:** Suppose R3 is a new address which needs to be inserted into the table, the hash function generates address as 110 for it. But this bucket is full to store the new data. In this case, a new bucket is inserted at the end of 110 buckets and is linked to it.



# **Dynamic Hashing**

* The dynamic hashing method is used to overcome the problems of static hashing like bucket overflow.
* In this method, data buckets grow or shrink as the records increases or decreases. This method is also known as Extendable hashing method.
* This method makes hashing dynamic, i.e., it allows insertion or deletion without resulting in poor performance.

## How to search a key

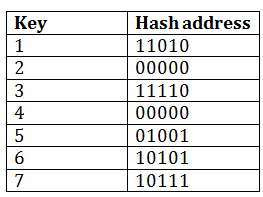
* First, calculate the hash address of the key.
* Check how many bits are used in the directory, and these bits are called as i.
* Take the least significant i bits of the hash address. This gives an index of the directory.
* Now using the index, go to the directory and find bucket address where the record might be.

## How to insert a new record

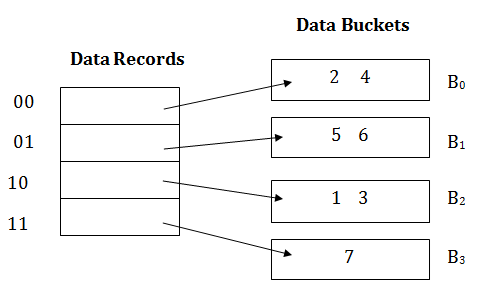
* Firstly, you have to follow the same procedure for retrieval, ending up in some bucket.
* If there is still space in that bucket, then place the record in it.
* If the bucket is full, then we will split the bucket and redistribute the records.

## For example:

Consider the following grouping of keys into buckets, depending on the prefix of their hash address:

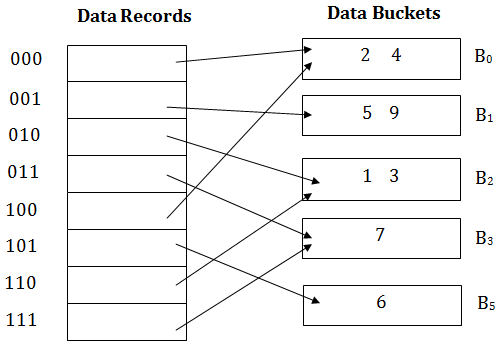


The last two bits of 2 and 4 are 00. So it will go into bucket B0. The last two bits of 5 and 6 are 01, so it will go into bucket B1. The last two bits of 1 and 3 are 10, so it will go into bucket B2. The last two bits of 7 are 11, so it will go into B3.



## Insert key 9 with hash address 10001 into the above structure:

* Since key 9 has hash address 10001, it must go into the first bucket. But bucket B1 is full, so it will get split.
* The splitting will separate 5, 9 from 6 since last three bits of 5, 9 are 001, so it will go into bucket B1, and the last three bits of 6 are 101, so it will go into bucket B5.
* Keys 2 and 4 are still in B0. The record in B0 pointed by the 000 and 100 entry because last two bits of both the entry are 00.
* Keys 1 and 3 are still in B2. The record in B2 pointed by the 010 and 110 entry because last two bits of both the entry are 10.
* Key 7 are still in B3. The record in B3 pointed by the 111 and 011 entry because last two bits of both the entry are 11.



## Advantages of dynamic hashing

* In this method, the performance does not decrease as the data grows in the system. It simply increases the size of memory to accommodate the data.
* In this method, memory is well utilized as it grows and shrinks with the data. There will not be any unused memory lying.
* This method is good for the dynamic database where data grows and shrinks frequently.

## Disadvantages of dynamic hashing

* In this method, if the data size increases then the bucket size is also increased. These addresses of data will be maintained in the bucket address table. This is because the data address will keep changing as buckets grow and shrink. If there is a huge increase in data, maintaining the bucket address table becomes tedious.
* In this case, the bucket overflow situation will also occur. But it might take little time to reach this situation than static hashing.