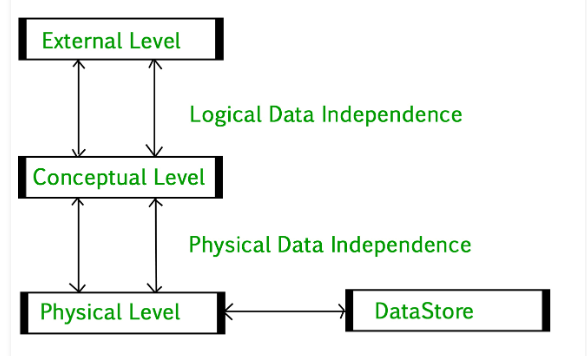
**DBMS QUESTION BANK (MID-I)**

**Module-1**

**1. Briefly explain the three tier architecture for the data independence**

**Ans:**

The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.DBMS 3-tier architecture divides the complete system into three inter-related but independent



1. **Physical Level:** At the physical level, the information about the location of database objects in the data store is kept. Various users of DBMS are unaware of the locations of these objects.In simple terms,physical level of a database describes how the data is being stored in secondary storage devices like disks and tapes and also gives insights on additional storage details.
2. **Conceptual Level:**At conceptual level, data is represented in the form of various database tables. For Example, STUDENT database may contain STUDENT and COURSE tables which will be visible to users but users are unaware of their storage.Also referred as logical schema,it describes what kind of data is to be stored in the database.
3. **External Level:** An external level specifies a view of the data in terms of conceptual level tables.  Each external level view is used to cater to the needs of a particular category of users. For Example, FACULTY of a university is interested in looking course details of students, STUDENTS are interested in looking at all details related to academics, accounts, courses and hostel details as well. So, different views can be generated for different users. The main focus of external level is data abstraction.

\*\*\*\* Data independence means a change of data at one level should not affect another level. Two types of data independence are present in this architecture:

1. **Physical Data Independence:** Any change in the physical location of tables and indexes should not affect the conceptual level or external view of data. This data independence is easy to achieve and implemented by most of the DBMS.
2. **Conceptual Data Independence:** The data at conceptual level schema and external level schema must be independent. This means a change in conceptual schema should not affect external schema. e.g.; Adding or deleting attributes of a table should not affect the user’s view of the table. But this type of independence is difficult to achieve as compared to physical data independence because the changes in conceptual schema are reflected in the user’s view.

**2. Explain advantages and disadvantages of DBMS over File System**

Ans:

|  |  |
| --- | --- |
| **DBMS** | **File System** |
| DBMS is a collection of data. In DBMS, the user is not required to write the procedures. | File system is a collection of data. In this system, the user has to write the procedures for managing the database. |
| DBMS gives an abstract view of data that hides the details. | File system provides the detail of the data representation and storage of data. |
| DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure. | File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost. |
| DBMS provides a good protection mechanism. | It is very difficult to protect a file under the file system. |
| DBMS contains a wide variety of sophisticated techniques to store and retrieve the data. | File system can't efficiently store and retrieve the data. |
| DBMS takes care of Concurrent access of data using some form of locking. | In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information. |

**3. Explain the concept of Specialization, generalization and aggregation in E\_R diagrams. Give one example for each one of them**

**Ans:**

**Generalization**

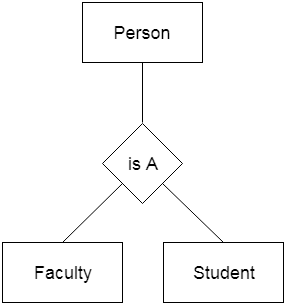
Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.

In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.

Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.

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

**For example,** Faculty and Student entities can be generalized and create a higher level entity Person.



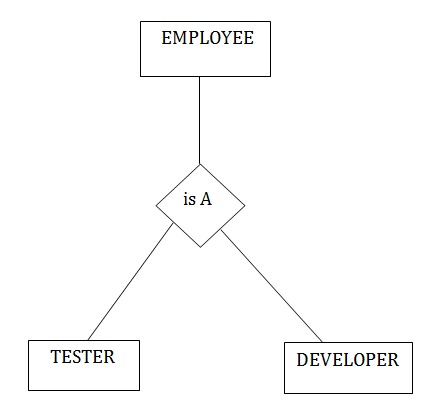
**Specialization**

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

Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.

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

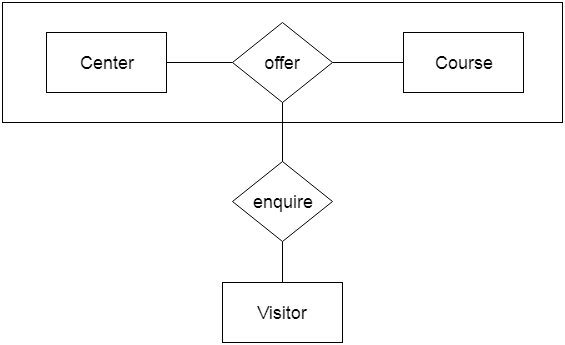
**For example:** In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



**Aggregation**

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

**For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



**4. List out various database applications and explain them.**

**Ans:**

**Applications of DBMS**

In so many fields, we will use a database management system.

Let’s see some of the applications where database management system uses −

* **Railway Reservation System** − The railway reservation system database plays a very important role by keeping record of ticket booking, train’s departure time and arrival status and also gives information regarding train late to people through the database.
* **Library Management System** − Now-a-days it’s become easy in the Library to track each book and maintain it because of the database. This happens because there are thousands of books in the library. It is very difficult to keep a record of all books in a copy or register. Now DBMS used to maintain all the information related to book issue dates, name of the book, author and availability of the book.
* **Banking** − Banking is one of the main applications of databases. We all know there will be a thousand transactions through banks daily and we are doing this without going to the bank. This is all possible just because of DBMS that manages all the bank transactions.
* **Universities and colleges** − Now-a-days examinations are done online. So, the universities and colleges are maintaining DBMS to store Student’s registrations details, results, courses and grade all the information in the database. For example, telecommunications. Without DBMS there is no telecommunication company. DBMS is most useful to these companies to store the call details and monthly postpaid bills.
* **Credit card transactions** − The purchase of items and transactions of credit cards are made possible only by DBMS. A credit card holder has to know the importance of their information that all are secured through DBMS.
* **Social Media Sites** − By filling the required details we are able to access social media platforms. Many users sign up daily on social websites such as Facebook, Pinterest and Instagram. All the information related to the users are stored and maintained with the help of DBMS.
* **Finance** − Now-a-days there are lots of things to do with finance like storing sales, holding information and finance statement management etc. these all can be done with database systems.
* **Military** − In military areas the DBMS is playing a vital role. Military keeps records of soldiers and it has so many files that should be kept secure and safe. DBMS provides a high security to military information.
* **Online Shopping** − Now-a-days we all do Online shopping without wasting the time by going shopping with the help of DBMS. The products are added and sold only with the help of DBMS like Purchase information, invoice bills and payment.
* **Human Resource Management** − The management keeps records of each employee’s salary, tax and work through DBMS.
* **Manufacturing** − Manufacturing companies make products and sell them on a daily basis. To keep records of all those details DBMS is used.
* **Airline Reservation system** − Just like the railway reservation system, airlines also need DBMS to keep records of flights arrival, departure and delay status.

So finally, we can clearly conclude that the DBMS is playing a very important role in each and every field.

**5. Explain centralized and client server architecture for the database**

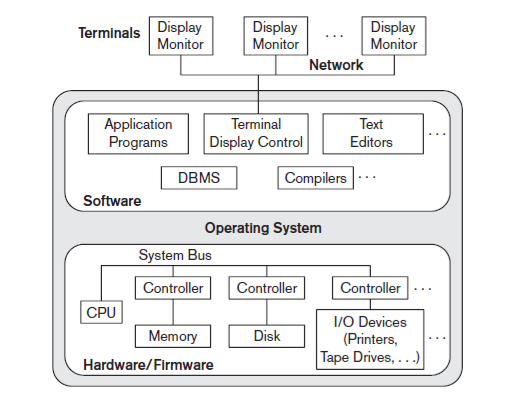
**Ans:**

# Centralized and Client/Server Architecture for DBMS

## Centralized Architecture for DBMS:

In Centralized Architecture, the mainframe computers are used for processing all system functions including User application Programs and User Interface Programs as well as DBMS functionalities. This is because in earlier days, most users accessed such systems via Computer Terminals, which can’t Process and they have only display capability. Therefore the Processing used to takeplace in these Computer Systems and the display information is sent to display terminals and these terminals are connected to mainframe computers via various kinds of Networks.

As the days pass by, we are now having PersonalComputers(PC’s) in the market. But still in the beginning Centralized Architecture for DBMS was used. Gradually the DBMS systems started to make use of Processing Power in the used side i.e Computers have come with Processing Power and in turn led to the use of Client/Server Architecture.



**Centralized Architecture for DBMS**

## Client/Server Architecture:

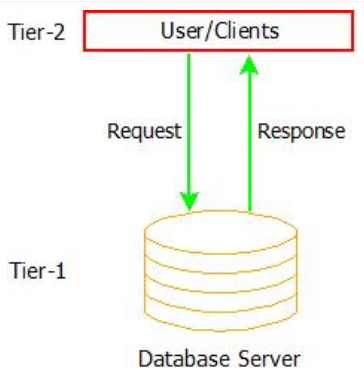
The concept of client/server architecture assumes an underlying framework that consists of many PCs as well as a smaller number of mainframe machines, connected via LANs and other types of computer networks.A client in  
this framework is typically a user machine that provides user interface capabilities and local processing. When a client requires access to additional functionality, such as database access which does not exist at that machine, it connects to a server that provides the needed functionality.

A Server is a system which contains both Hardware and Software which provides services to client Machines like file access, printing and database access.

**1)Two Tier Client/Server Architecture for DBMS:**

Here Two-tier means that our Architecture has two layers, which are client layer and Datalayer. In Client layer we have several Client machines which can have the access to the database server. The API present on the client machine will establish the connection between the machine and the Database server through JDBC somthing else.This is because Clients and Database Server may be at different different locations.Once this connection gets established,the Interface present on the client machine contains an Application Program on the back-side which contains a query. This query will be processed by the Database server and in turn the queried information will be sent to the client machine.

For example if we query the database to retrieve some information, the query will be Processed by Database server and that information will be sent to the client by Database server itself!!!

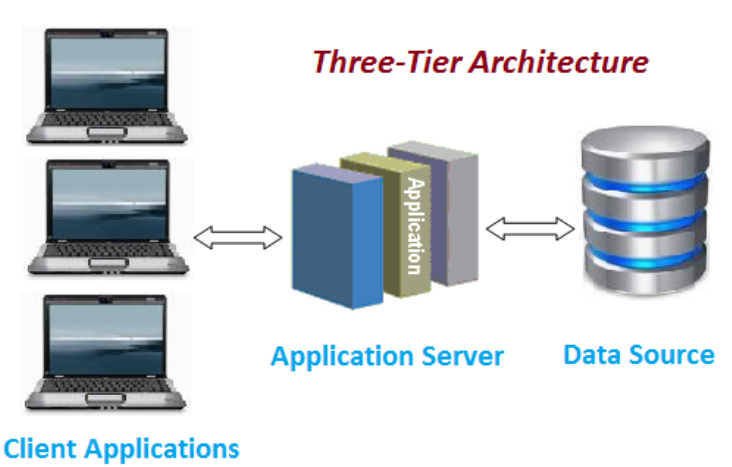


**Two-Tier Architecture for DBMS**

**2) Three-Tier client/server Architecture for DBMS:**

Here there is an additional layer which acts as an intermediate between Client layer and Datalayer called Business logic layer. Business logic layer is the layer where the Application Programs are processed. Here the Application Programs are processed in the Application server itself, which makes it different from Two-tier Architecture where queries are processed in the database server.

Simply the Client machines will contact Application Server which in turn processes our Application Programs and fetches the Required Data from Database and then sends this Information back to the client machine in the suitable format only.



**Three-Tier client/server Architecture for DBMS**

Now we may think that Two-Tier Architecture is easy to use and maintain and why we should go for Three-Tier. The Reason is Three-Tier Architecture is Scalable and more secured.

Even it is easy to maintain Two-Tier Architecture of DBMS it is still not scalable when we have large number of clients and also not secure because the clients are having direct access to database server. But Three-Tier Architecture ensures Scalability and Security of the data because of the presence of this Intermediate layer which processes the queries and it just retrieves data from server instead of processing in the server to take place.

**6. What is degree of relation and mapping cardinality and explain various relations with ER representation**

**Ans:**

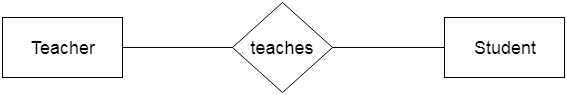
In DBMS, a degree of relationship **represents the number of entity types that associate in a relationship**. For example, we have two entities, one is a student and the other is a bag and they are connected with the primary key and foreign key.

Mapping Cardinalities or Cardinality Ratios:

**It tells the number of entities to with another entity can be associated through a 'relationship set'** . Mapping cardinality uses binary relationship sets. So a binary relationship set R between entity set A and B there will be one of this mapping cardinality.

**Various Relationship with Er**

* A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

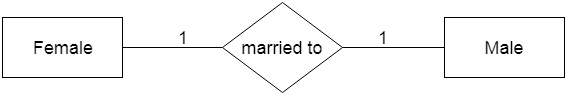


Types of relationship are as follows:

**One-to-One Relationship**

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

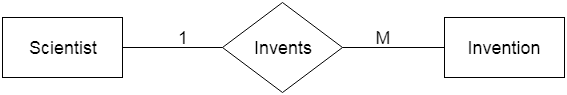
**For example,** A female can marry to one male, and a male can marry to one female.



**One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

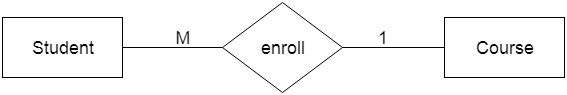
**For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



**Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

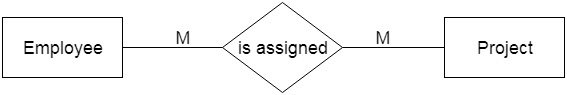
**For example,** Student enrolls for only one course, but a course can have many students.



**Many-to-one relationship**

When more than one instance of the entity on the left, and many instances of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

**For example,** Many Employees assigned to many Projects.



**7. Explain the difference between the weak entity and strong entity with ER diagram representation**

**Ans:**

**Entity:**

* An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
* Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

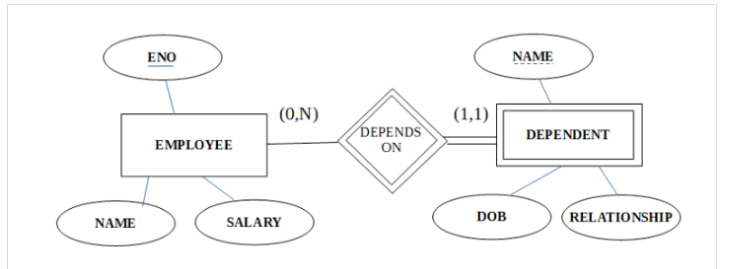
Weak entity:

A weak entity cannot be used independently as it is dependent on a strong entity type known as its owner entity. Also, the relationship that connects the weak entity to its owner identity is called the identifying relationship.

A weak entity always has a total participation constraint with respect to its identifying relationship because it cannot be identified independently of its owner identity.

A weak entity may have a partial key, which is a list of attributes that identify weak entities related to the same owner entity.

In the ER diagram, both the weak entity and its corresponding relationship are represented using a double line and the partial key is underlined with a dotted line.



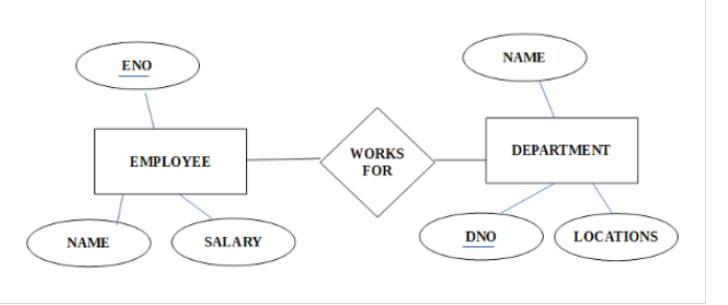
In the given ER diagram, Dependent is the weak entity and it depends on the strong entity Employee via the relationship Depends on.

There can be an employee without a dependent in the Company but there will be no record of the Dependent in the company systems unless the dependent is associated with an Employee.

**Strong entity:**

A strong entity is complete by itself and is not dependent on any other entity type. It possess a primary key which describes each instance in the strong entity set uniquely. That means any element in the strong entity set can be uniquely identified.

A Strong entity is represented by a square with a single line unlike a Weak Entity which contained double lines.



**8. Define and discuss the functions of Database Administrator (DBA)**

**Ans:**

**Database administration** is more of an operational or technical level function responsible for physical database design, security enforcement, and database performance.  Tasks include maintaining the data dictionary, monitoring performance, and enforcing organizational standards and security.

These are the functions of a data administrator (not to be confused with database administrator functions):

1. Data policies, procedures, standards

2. Planning- development of organization's IT strategy, enterprise model, cost/benefit model, design of database environment, and administration plan.

3. Data conflict (ownership) resolution

4. Data analysis- Define and model data requirements, business rules, operational requirements, and maintain corporate data dictionary

5. Internal marketing of DA concepts

6. Managing the data repository

**The DBA is responsible for many critical tasks:**

**Design of the conceptual and physical schemas:** The DBA is responsible for interacting with the users of the system to understand what data is to be stored in the DBMS and how it is likely to be used. Based on this knowledge, the DBA must design the conceptual schema (decide what relations to store) and the physical schema (decide how to store them). The DBA may also design widely used portions of the external schema, although users will probably augment this schema by creating additional views.

**Security and authorization:** The DBA is responsible for ensuring that unauthorized data access is not permitted. In general, not everyone should be able to access all the data. In a relational DBMS, users can be granted permission to access only certain views and relations. For example, although you might allow students to find out course enrollments and who teaches a given course, you would not want students to see faculty salaries or each others' grade information.

**Data availability and recovery from failures:** The DBA must take steps to ensure that if the system fails, users can continue to access as much of the uncorrupted data as possible. The DBA must also work to restore the data to a consistent state. The DBMS provides software support for these functions, but the DBA is responsible for implementing procedures to back up the data periodically and to maintain logs of system activity (to facilitate recovery from a crash).

**Database tuning:** The needs of users are likely to evolve with time. The DBA is responsible for modifying the database, in particular the conceptual and physicalschemas, to ensure adequate performance as user requirements change.

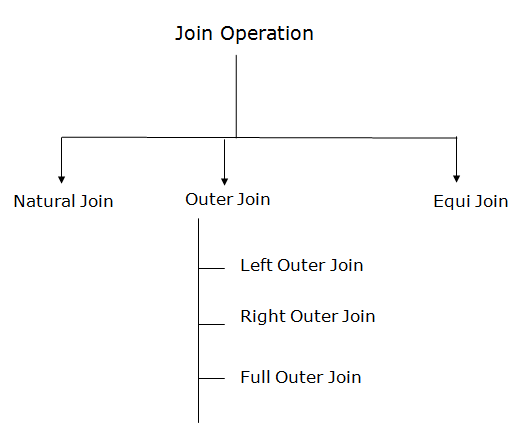
**Module-2**

**1. Classify different join operations (Relational Algebra& SQL) and explain with example.**

**Ans:**

**Join Operations:**

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈. Example: Operation: (EMPLOYEE ⋈ SALARY)



**Joins in Relational Algebra**

1. **Natural Join:**

A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.

It is denoted by ⋈.

**Example:** Let's use the above EMPLOYEE table and SALARY table:

∏EMP\_NAME, SALARY (EMPLOYEE ⋈ SALARY)

2. **Outer Join:**

The outer join operation is an extension of the join operation. It is used to deal with missing information.

**Example:** (EMPLOYEE ⋈ FACT\_WORKERS)

An outer join is basically of three types:

1.Left outer join

2.Right outer join

3.Full outer join

a. **Left outer join:**

Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.

In the left outer join, tuples in R have no matching tuples in S. It is denoted by ⟕.

**Example:** Using the above EMPLOYEE table and FACT\_WORKERS table

EMPLOYEE ⟕ FACT\_WORKERS

b. **Right outer join:**

Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.

In right outer join, tuples in S have no matching tuples in R.It is denoted by ⟖.

**Example:** Using the above EMPLOYEE table and FACT\_WORKERS Relation

EMPLOYEE ⟖ FACT\_WORKERS

c. **Full outer join:**

Full outer join is like a left or right join except that it contains all rows from both tables.

In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name. It is denoted by ⟗.

**Example:** Using the above EMPLOYEE table and FACT\_WORKERS table

EMPLOYEE ⟗ FACT\_WORKERS

3. **Equi join:**

It is also known as an inner join. It is the most common join. It is based on matched data as per the equality condition. The equi join uses the comparison operator(=).

**Example:** CUSTOMER ⋈ PRODUCT

# SQL Joins

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

example :

we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables: Orders, Customers

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

SQL INNER JOIN Keyword

The INNER JOIN keyword selects records that have matching values in both tables.



The following SQL statement selects all orders with customer information:

### Example

SELECT Orders.OrderID, Customers.CustomerName  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

SQL LEFT JOIN Keyword

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.



The following SQL statement will select all customers, and any orders they might have:

### Example

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;

SQL RIGHT JOIN Keyword

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.



The following SQL statement will return all employees, and any orders they might have placed:

### Example

SELECT Orders.OrderID, Employees.LastName, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
ORDER BY Orders.OrderID;

SQL FULL OUTER JOIN Keyword

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.



The following SQL statement selects all customers, and all orders:

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;

SQL Self Join

A self join is a regular join, but the table is joined with itself.

The following SQL statement matches customers that are from the same city:

### Example

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID  
AND A.City = B.City  
ORDER BY A.City;

**2. What is view of data? Explain the three levels of data independence.**

**Ans:**

View of data in DBMS **describes the abstraction of data at three-level i.e. physical level, logical level, view level**. The physical level of abstraction defines how data is stored in the storage and also reveals its access path

**Data Abstraction** is a process of hiding unwanted or irrelevant details from the end user.

It provides a different **view** and helps in achieving data independence which is used to enhance the security of data.

The database systems consist of complicated data structures and relations. For users to access the data easily, these complications are kept hidden, and only the relevant part of the database is made accessible to the users through data abstraction.

Mainly there are three levels of abstraction for DBMS, which are as follows −

* Physical or Internal Level
* Logical or Conceptual Level
* View or External Level



**Physical or Internal Level**

It is the lowest level of abstraction for DBMS which defines how the data is actually stored, it defines data-structures to store data and access methods used by the database. Actually, it is decided by developers or database application programmers how to store the data in the database.

So, overall, the entire database is described in this level that is physical or internal level. It is a very complex level to understand. For example, customer's information is stored in tables and data is stored in the form of blocks of storage such as bytes, gigabytes etc.

**Logical or Conceptual Level**

Logical level is the intermediate level or next higher level. It describes what data is stored in the database and what relationship exists among those data. It tries to describe the entire or whole data because it describes what tables to be created and what are the links among those tables that are created.

It is less complex than the physical level. Logical level is used by developers or database administrators (DBA). So, overall, the logical level contains tables (fields and attributes) and relationships among table attributes.

**View or External Level**

It is the highest level. In view level, there are different levels of views and every view only defines a part of the entire data. It also simplifies interaction with the user and it provides many views or multiple views of the same database.

View level can be used by all users (all levels' users). This level is the least complex and easy to understand.

For example, a user can interact with a system using GUI that is view level and can enter details at GUI or screen and the user does not know how data is stored and what data is stored, this detail is hidden from the user.

# Data Independence

* Data independence can be explained using the three-schema architecture.
* Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

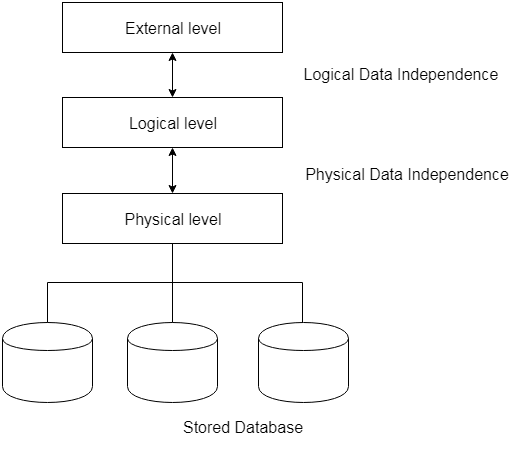
There are two types of data independence:

## 1. Logical Data Independence

* Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
* Logical data independence is used to separate the external level from the conceptual view.
* If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.
* Logical data independence occurs at the user interface level.

## 2. Physical Data Independence

* Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
* If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
* Physical data independence is used to separate conceptual levels from the internal levels.
* Physical data independence occurs at the logical interface level.



**3. Explain following operators in Relational Algebra i) selection ii) projection iii) rename**

**Ans:**

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

1. **Select Operation:**

The select operation selects tuples that satisfy a given predicate.It is denoted by sigma (σ). Notation:  σ p(r)

**Where:**

* **σ** is used for selection prediction  
  **r** is used for relation  
  **p** is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

Example :

σ BRANCH\_NAME="perryride" (LOAN)

2. **Project Operation:**

This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.

It is denoted by ∏. Notation: ∏ A1, A2, An (r)

**Where** **A1**, **A2**, **A3** is used as an attribute name of relation **r**.

**Example: CUSTOMER RELATION**

∏ NAME, CITY (CUSTOMER)

**3. Rename Operation:**

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

**Example:** We can use the rename operator to rename STUDENT relation to STUDENT1.

ρ(STUDENT1, STUDENT)

**4. Define Relational Model & Explain the concept of domain, attribute, tuple, relation with an example?**

**Ans:**

**Relational Model concept**

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

Relational Model (RM) represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

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

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

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

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

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

Tuple – It is nothing but a single row of a table, which contains a single record.

Properties of Relations

* Name of the relation is distinct from all other relations.
* Each relation cell contains exactly one atomic (single) value
* Each attribute contains a distinct name
* Attribute domain has no significance
* tuple has no duplicate value
* Order of tuple can have a different sequence

**Example: STUDENT Relation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NAME** | **ROLL\_NO** | **PHONE\_NO** | **ADDRESS** | **AGE** |
| Ram | 14795 | 7305758992 | Noida | 24 |
| Shyam | 12839 | 9026288936 | Delhi | 35 |
| Laxman | 33289 | 8583287182 | Gurugram | 20 |
| Mahesh | 27857 | 7086819134 | Ghaziabad | 27 |
| Ganesh | 17282 | 9028 9i3988 | Delhi | 40 |

In the given table, NAME, ROLL\_NO, PHONE\_NO, ADDRESS, and AGE are the attributes.

The instance of schema STUDENT has 5 tuples.

t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

**5. Explain Commit, Rollback and Save point commands in SQL with suitable examples**

**Ans:**

**TCL Commands in SQL**

**In SQL, TCL stands for Transaction control language.**

**A single unit of work in a database is formed after the consecutive execution of commands is known as a transaction.**

**There are certain commands present in SQL known as TCL commands that help the user manage the transactions that take place in a database.**

**COMMIT. ROLLBACK and SAVEPOINT are the most commonly used TCL commands in SQL.**

**1. COMMIT**

**COMMIT command in SQL is used to save all the transaction-related changes permanently to the disk.**

**Syntax:COMMIT;**

**Example**

Consider the CUSTOMERS table having the following records −

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

Following is an example which would delete those records from the table which have age = 25 and then COMMIT the changes in the database.

SQL> DELETE FROM CUSTOMERS

WHERE AGE = 25;

SQL> COMMIT;

Thus, two rows from the table would be deleted and the SELECT statement would produce the following result.

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

**2. SAVEPOINT**

**We can divide the database operations into parts. For example, we can consider all the insert related queries that we will execute consecutively as one part of the transaction and the delete command as the other part of the transaction. Using the SAVEPOINT command in SQL, we can save these different parts of the same transaction using different names. *For example*, we can save all the insert related queries with the savepoint named INS. To save all the insert related queries in one savepoint, we have to execute the SAVEPOINT query followed by the savepoint name after finishing the insert command execution.**

**Syntax:**

**SAVEPOINT savepoint\_name;**

**Example**

Consider the CUSTOMERS table having the following records.

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

The following code block contains the series of operations.

SQL> SAVEPOINT SP1;

Savepoint created.

SQL> DELETE FROM CUSTOMERS WHERE ID=1;

1 row deleted.

SQL> SAVEPOINT SP2;

Savepoint created.

SQL> DELETE FROM CUSTOMERS WHERE ID=2;

1 row deleted.

SQL> SAVEPOINT SP3;

Savepoint created.

SQL> DELETE FROM CUSTOMERS WHERE ID=3;

1 row deleted.

Now that the three deletions have taken place, let us assume that you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP2. Because SP2 was created after the first deletion, the last two deletions are undone −

SQL> ROLLBACK TO SP2;

Rollback complete.

Notice that only the first deletion took place since you rolled back to SP2.

SQL> SELECT \* FROM CUSTOMERS;

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

6 rows selected.

**3. ROLLBACK**

**While carrying a transaction, we must create savepoints to save different parts of the transaction. According to the user's changing requirements, he/she can roll back the transaction to different savepoints. *Consider a scenario*: We have initiated a transaction followed by the table creation and record insertion into the table. After inserting records, we have created a savepoint INS. Then we executed a delete query, but later we thought that mistakenly we had removed the useful record. Therefore in such situations, we have an option of rolling back our transaction. In this case, we have to roll back our transaction using the *ROLLBACK* command to the savepoint INS, which we have created before executing the DELETE query.**

**Syntax:**

**ROLLBACK TO savepoint\_name;**

**Example**

Consider the CUSTOMERS table having the following records −

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

Following is an example, which would delete those records from the table which have the age = 25 and then ROLLBACK the changes in the database.

SQL> DELETE FROM CUSTOMERS

WHERE AGE = 25;

SQL> ROLLBACK;

Thus, the delete operation would not impact the table and the SELECT statement would produce the following result.

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

**6. What are the SQL constructs to modify the structure of tables and destroy tables**

**Ans:**

The SQL CONSTRUCT QUERY statement **constructs an SQL statement text string from a list of data items and literal values**. A directive that specifies formatting may optionally precede each data item or literal value argument. SqlTextString (output). This argument must refer to a nonnumeric data item.

## The SQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

### Syntax

CREATE TABLE table\_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
   ....  
);

### Example

CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);

## SQL ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

## ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

ALTER TABLE table\_name  
ADD column\_name datatype;

The following SQL adds an "Email" column to the "Customers" table:

### Example

ALTER TABLE Customers  
ADD Email varchar(255);

## The SQL DROP TABLE Statement

The DROP TABLE statement is used to drop an existing table in a database.

### Syntax

DROP TABLE table\_name;

### Example

DROP TABLE Shippers;

**7. Discussed what happens when Views are destroyed**

**Ans:**

**Views in SQL**

Views in SQL are considered as a virtual table. A view also contains rows and columns.

To create the view, we can select the fields from one or more tables present in the database.

A view can either have specific rows based on certain condition or all the rows of a table.

Some Views are used only for looking at table data. Other Views can be used to Insert, Update and Delete table data as well as View data. If a View is used to only look at table data and nothing else the View is called a Read-Only View. A View that is used to look at table data as well as Insert, Update and Delete table data is called an Updateable View.   
The reasons why views are created are:

* When Data security is required .
* When Data redundancy is to be kept to the minimum while maintaining data security .

**Creating view**

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

**Syntax:**

CREATE VIEW view\_name AS  SELECT column1, column2.....

FROM table\_name  WHERE condition;

**Query:**

CREATE VIEW DetailsView AS  SELECT NAME, ADDRESS

FROM Student\_Details  WHERE STU\_ID < 4;

**Deleting View**

A view can be deleted using the Drop View statement.

**Syntax :**DROP VIEW view\_name;

**Example:**

DROP VIEW MarksView;

**After deleting the view the data in the table referred by the view won't be deleted.**

**8. What is Relation? Differentiate between a relation Schema and Relation Instance**

**Ans:**

A relationship in a DBMS, is **primarily the way two or more data sets are linked**. This is so true for Relational Database Management Systems. One dataset may be then termed as the Foreign key and the ones linked to it may be termed as the Primary Key. There may be multiple Foreign and Primary keys linked to each other.

**Relational Model concept**

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

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

**example: STUDENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NAME** | **ROLL\_NO** | **PHONE\_NO** | **ADDRESS** | **AGE** |
| Ram | 14795 | 7305758992 | Noida | 24 |
| Shyam | 12839 | 9026288936 | Delhi | 35 |
| Laxman | 33289 | 8583287182 | Gurugram | 20 |
| Mahesh | 27857 | 7086819134 | Ghaziabad | 27 |
| Ganesh | 17282 | 9028 9i3988 | Delhi | 40 |

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

A relational schema is an outline that shows how companies store and organize information within a database. It also shows what connections make up the database. Developers often view relational schemas as the shape, blueprint or design of the sets of information within the database. Relational schemas do not contain actual data, however, because it is simply a blueprint. A developer's goal is to design the schema in a way that the information is readable and avoids redundancy. The developer can choose to display a schema as a visual depiction, like a graph, or as formulas written in coding language.

**ex : STUDENT(name,roll\_no,phone\_no,address,age)**

**Module-3**

**1. Discuss in detail about PL/SQL Procedures with examples**

**Ans:**

**PL/SQL Concepts**

What is PL/SQL

PL/SQL is a block structured language. The programs of PL/SQL are logical blocks that can contain any number of nested sub-blocks. Pl/SQL stands for "Procedural Language extension of SQL" that is used in Oracle. PL/SQL is integrated with Oracle database (since version 7). The functionalities of PL/SQL usually extended after each release of Oracle database. Although PL/SQL is closely integrated with SQL language, yet it adds some programming constraints that are not available in SQL.

**Block Structure**

* PL/SQL blocks have a pre-defined structure in which the code is to be grouped. Below are different sections of PL/SQL blocks.

**1. Declaration section**

**2. Execution section**

**3. Exception-Handling section**

syntax of pl/sql structure :

**DECLARE –optional**

**<declarations>**

**BEGIN --mandatory**

**<executable statements. At least one executable statement is mandatory>**

**EXCEPTION --optional**

**<exception handles>**

**END; --mandatory**

**/**

Example of PL/SQL constant

Let's take an example to explain it well:

**DECLARE**

   -- constant declaration

   pi constant number := 3.141592654;

   -- other declarations

   radius number(5,2);      dia number(5,2);      circumference number(7, 2);     area number (10, 2);

**BEGIN**

   -- processing

   radius := 9.5;

   dia := radius \* 2;      circumference := 2.0 \* pi \* radius;     area := pi \* radius \* radius;

   -- output

   dbms\_output.put\_line('Radius: ' || radius);

   dbms\_output.put\_line('Diameter: ' || dia);

   dbms\_output.put\_line('Circumference: ' || circumference);

   dbms\_output.put\_line('Area: ' || area);

**END**;

/

After the execution of the above code at SQL prompt, it will produce the following result:.

Radius: 9.5

Diameter: 19

Circumference: 59.69

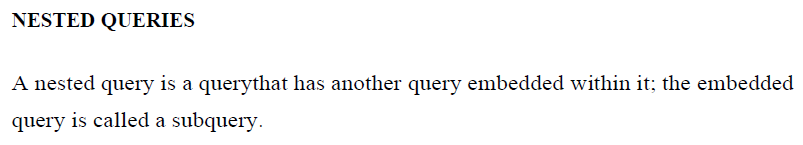
Area: 283.53

Pl/SQL **procedure** successfully completed.

PL/SQL L

**2. Discuss about Nested queries with an example.**

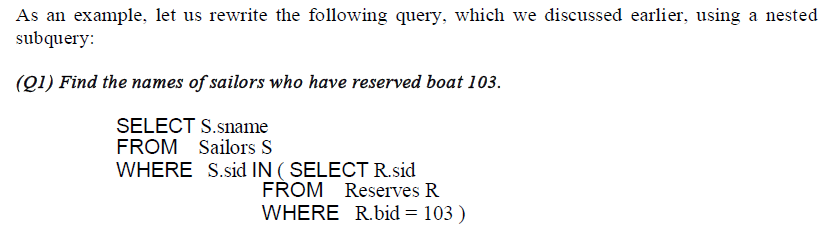
**Ans:**

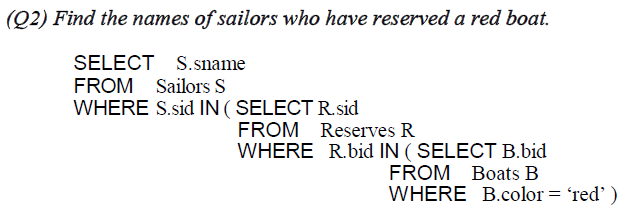


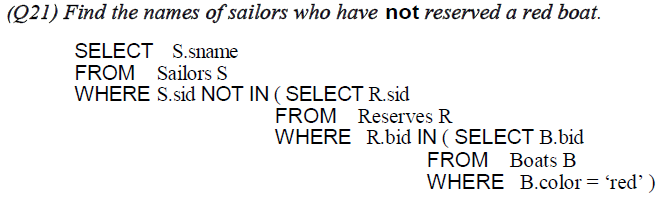
SQL subqueries are a powerful tool. They allow us to perform tasks more efficiently by having only one query instead of several.

When using nested queries, **keep these** **considerations in mind**:

* Subqueries can return **single values or tables** (with one or many rows and columns).
* You can include a subquery:
  + In the **WHERE** clause, to filter data.
  + In the **FROM** clause, to specify a new table.
  + In the **SELECT** clause, to specify a certain column.
  + In the **HAVING** clause, as a group selector.
* Subqueries should always be enclosed in parentheses ().
* Different database management systems have certain **limitations** on the number of subquery levels (e.g. up to 32 levels in SQL Server). However, in practice, you’ll rarely have more than 2-3 levels of nested queries.
* Subqueries are often **computationally inefficient**. Thus, I recommend avoiding nested queries when other options are available (e.g. JOINs).





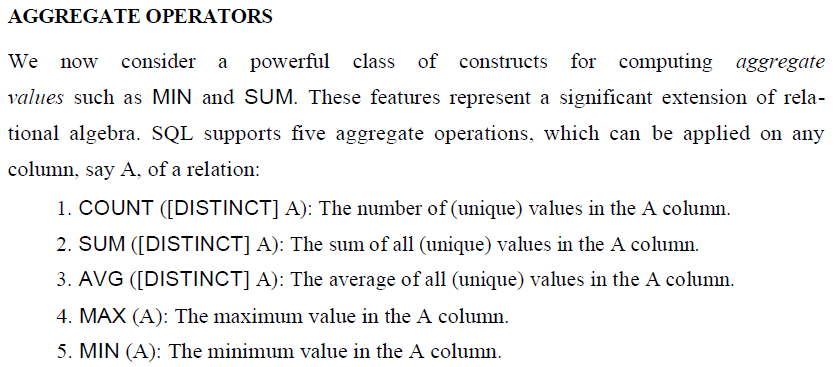


First of all, you can put a nested SELECT **within the WHERE clause** with comparison operators or the IN, NOT IN, ANY, or ALL operators. The second group of operators are used when your subquery returns a list of values (rather than a single value, as in the previous example):

* The IN operator checks if a certain value **is in the table** returned by the subquery.
* The NOT IN operator filters out the rows corresponding to the values **not present** in that table returned by a subquery.
* The ANY operator is used with comparison operators to **evaluate if any of the values** returned by the subquery satisfy the condition.
* The ALL operator is also used with comparison operators to **evaluate if all values** returned by the subquery satisfy the condition.

**3. Discuss about different types of aggregate operators in SQL with examples?**

**Ans:**

****

**1. COUNT()**

**COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.**

**COUNT function uses the COUNT(\*) that returns the count of all the rows in a specified table. COUNT(\*) considers duplicate and Null.**

**Syntax : COUNT(\*)  or  COUNT( [ALL|DISTINCT] expression )**

**ex: SELECT COUNT(\*)  FROM PRODUCT\_MAST;**

**Output: 10**

**Example: COUNT with WHERE**

**SELECT COUNT(\*)  FROM PRODUCT\_MAST WHERE RATE>=20;**

**Output: 7**

**Example: COUNT() with GROUP BY**

**SELECT COMPANY, COUNT(\*)   FROM PRODUCT\_MAST  GROUP BY COMPANY;**

**Output:**

**Com1 5**

**Com2 3**

**Com3 2**

**Example: COUNT() with HAVING**

**SELECT COMPANY, COUNT(\*)  FROM PRODUCT\_MAST  GROUP BY COMPANY  HAVING COUNT(\*)>2;**

**Output:**

**Com1 5**

**Com2 3**

**2. SUM()**

**Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.**

**Syntax : SUM()  or  SUM( [ALL|DISTINCT] expression )**

**Example: SUM()**

**SELECT SUM(COST)  FROM PRODUCT\_MAST;**

**Output:**

**670**

**3. AVG ()**

**The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.**

**Syntax**

**AVG()  or  AVG( [ALL|DISTINCT] expression )**

**Example:**

**SELECT AVG(COST)  FROM PRODUCT\_MAST;**

**Output:**

**67.00**

**4. MAX()**

**MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.**

**Syntax : MAX()  or  MAX( [ALL|DISTINCT] expression )**

**Example:**

**SELECT MAX(RATE)  FROM PRODUCT\_MAST;**

**Output :**

**30 5.**

**5. MIN()**

**MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.**

**Syntax : MIN()  or  MIN( [ALL|DISTINCT] expression )**

**Example:**

**SELECT MIN(RATE)  FROM PRODUCT\_MAST;**

**Output: 10**

**4. Explain the term stored procedure and give examples why stored procedure are useful**

**PL/SQL Procedure**

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

**Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.

**Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

How to pass parameters in procedure:

When you want to create a procedure or function, you have to define parameters .There is three ways to pass parameters in procedure:

**IN parameters:**The IN parameter can be referenced by the procedure or function. The value of the parameter cannot be overwritten by the procedure or the function.

**OUT parameters:**The OUT parameter cannot be referenced by the procedure or function, but the value of the parameter can be overwritten by the procedure or function.

**INOUT parameters:**The INOUT parameter can be referenced by the procedure or function and the value of the parameter can be overwritten by the procedure or function.

A procedure may or may not return any value.

**Create procedure example**

**In this example, we are going to insert record in user table. So you need to create user table first.**

**Table creation:**

**create table user(id number(10) primary key,name varchar2(100));**

**Now write the procedure code to insert record in user table.**

**Procedure Code:**

**create or replace procedure "INSERTUSER"  (id IN NUMBER,  name IN VARCHAR2)**

**is**

**begin**

**insert into user values(id,name);**

**end;**

**/**

**Output:**

**Procedure created.**

A stored procedure **provides an important layer of security between the user interface and the database**. It supports security through data access controls because end users may enter or change data, but do not write procedures.

**The main advantages of stored procedure are given below:**

* Better Performance – The procedure calls are quick and efficient as stored procedures are compiled once and stored in executable form.
* Higher Productivity
* Ease of Use
* Scalability
* Maintainability
* Security