S.A. ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University)

QUESTION BANK

EVEN SEMESTER, 2023-2024

SUBJECT CODE: IT1401A

SUBJECT TITLE: DATABASE MANAGEMENT SYSTEMS

SEM/YEAR: IV SEM/II YEAR

UNIT I - RELATIONAL DATABASES
Purpose of Database System – Views of data – Data Models – Database System Architecture –Introduction to relational databases – Relational Model – Keys – Relational Algebra – SQL fundamentals – Advanced SQL features – Embedded SQL– Dynamic SQL

PART-A

1. **List the disadvantages of file system over database.**

   Here are some disadvantages of using a file system compared to a database:

   - ✓ Data Redundancy
   - ✓ Data Integrity
   - ✓ Limited Query Capabilities
   - ✓ Concurrency Control
   - ✓ Data Atomicity and Durability
   - ✓ Backup and Recovery.

2. **Mention the major responsibilities of a database administrator.**

   Some major responsibilities of a database administrator (DBA):

   - ✓ Database Installation and Configuration
   - ✓ Database Design and Schema Management
   - ✓ Data Security and Access Control
   - ✓ Backup and Recovery
   - ✓ Performance Monitoring and Tuning.

3. **State the levels of abstraction in a DBMS.**
   In a Database Management System (DBMS), there are typically three levels of abstraction:
   - ✓ External Level (View Level)
   - ✓ Conceptual Level (Logical Level)
   - ✓ Internal Level (Physical Level)

4. **What is a data model? List the types of data model**

   A data model is a conceptual representation of how data is organized and structured within a database system. It defines the logical relationships between different data elements and the rules governing those relationships. Data models serve as a blueprint for designing databases, providing a framework for understanding and manipulating data.

   Here are the types of data models commonly used in database design:

   - ✓ Hierarchical Data Model
   - ✓ Network Data Model
   - ✓ Relational Data Model
   - ✓ Entity-Relationship (ER) Model
   - ✓ Object-Oriented Data Model
   - ✓ Document Data Model

5. **Define foreign key. Give an example.**

   A foreign key is a column or a set of columns in a relational database table that provides a link between data in two tables. It establishes a relationship between two tables by referencing the primary key or a unique key in another table. The foreign key constraint ensures referential integrity, meaning that values in the foreign key column(s) must match values in the referenced primary key or unique key column(s) in the related table.

   Here's an example to illustrate the concept of a foreign key:

   Consider two tables: **Orders** and **Customers**

- Columns: OrderID (Primary Key), CustomerID (Foreign Key), OrderDate, TotalAmount, etc.
- Each row represents an order placed by a customer.
- The **CustomerID** column in the **Orders** table is a foreign key referencing the **CustomerID** column in the **Customers** table.

6. **Write a relational algebra expression for selecting all tuples from the relation Loan with the condition Loan amount greater than 5000 rupees.**

   Relational algebra expression for selecting all tuples from the relation Loan with the condition Loan amount greater than 5000 rupees:

   $$\sigma(\text{LoanAmount} > 5000)(\text{Loan}).$$

7. **Differentiate primary key from candidate key.**

   - Primary key: A primary key is a column or set of columns that uniquely identifies each row in a table. It is chosen from among the candidate keys and is used to enforce entity integrity.

   - Candidate key: A candidate key is a column or a set of columns in a table that can uniquely identify each row in the table. There can be multiple candidate keys in a table.

8. What are aggregate functions? List some aggregate functions Supported by SQL.

Aggregate functions are functions in SQL used to perform a calculation on a set of values and return a single value. Some aggregate functions supported by SQL include:

- SUM

- AVG
- COUNT
- MAX
- MIN

9. Why does SQL allow duplicate tuples in a table or in a query result?

SQL allows duplicate tuples in a table or query result because it allows for storing and manipulating data in a flexible manner, accommodating various use cases and data requirements.

10. Differentiate between dynamic SQL and static SQL.

- Dynamic SQL: SQL statements are constructed and executed at runtime. It allows for dynamic modification of the SQL statements based on program conditions.

- Static SQL: SQL statements are predefined and compiled into an application before execution. Changes to SQL statements require modification and recompilation of the application code.

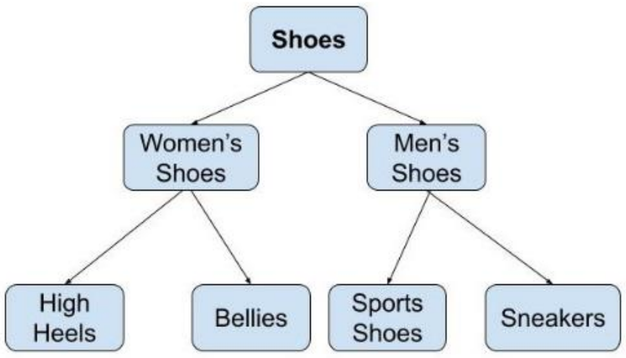
PART-B

1. Compare and contrast various data model groups, providing relevant examples for each.

Data Models

The Data Model is a collection of conceptual tools for describing data, data relationship, data semantics, and consistency constraints. A Data Model is a logical structure of Database. A data model provides a way to describe the design of a database at the physical, logical and view level. The purpose of a data model is to represent data and make the data understandable. Data Models are used to show how data is stored, connected, accessed and updated in the database management system. Though there are many data models being used nowadays but the Relational model is the most widely used model. Some of the Data Models in DBMS are:

1. Hierarchical Model: Hierarchical Model was the first DBMS model. This model organizes the data in the hierarchical tree structure. The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node. This model easily represents some of the real-world relationships like food recipes, sitemap of a website etc.



Hierarchical Model

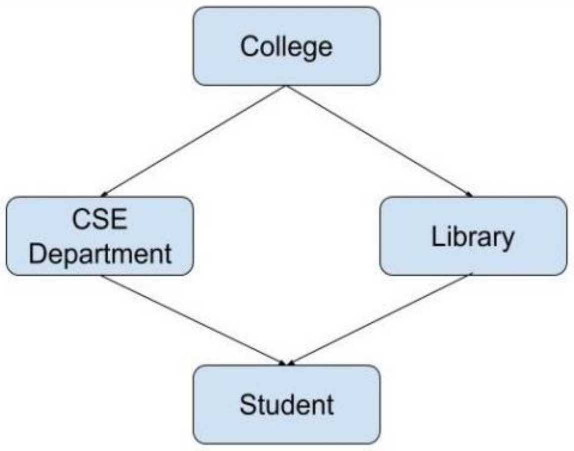
Advantages of Hierarchical Model:

- It is very simple and fast to traverse through a tree-like structure.
- Any change in the parent node is automatically reflected in the child node so, the integrity of data is maintained.

Disadvantages of Hierarchical Model:

- Complex relationships are not supported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent node then that can't be represented using this model.
- If a parent node is deleted then the child node is automatically deleted.

2. Network Model: This model is an extension of the hierarchical model. It was the most popular model before the relational model. This model is the same as the hierarchical model, the only difference is that a record can have more than one parent. It replaces the hierarchical tree with a graph.



Network Model

Advantages of Network Model

- The data can be accessed faster as compared to the hierarchical model. This is because the data is more related in the network model and there can be more than one path to reach a particular node. So the data can be accessed in many ways.
- As there is a parent-child relationship so data integrity is present. Any change in parent record is reflected in the child record.

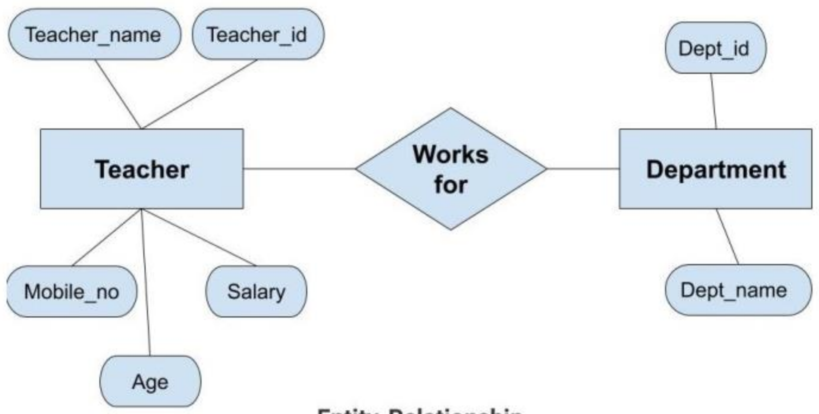
Disadvantages of Network Model

- As more and more relationships need to be handled the system might get complex. So, a user must be having detailed knowledge of the model to work with the model.
- Any change like updation, deletion, insertion is very complex.

3. Entity-Relationship Model

Entity-Relationship Model or simply ER Model is a high-level data model diagram. In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand. It is also very easy for the developers to understand the system by just looking at the ER diagram. ER diagram has the following three components:

- **Entities**: Entity is a real-world thing. It can be a person, place, or even a concept. Example: Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.
- **Attributes**: An entity contains a real-world property called attribute. This is the characteristics of that attribute. Example: The entity teacher has the property like teacher id, salary, age, etc.
- **Relationship**: Relationship tells how two attributes are related. Example: Teacher works for a department.



Entity-Relationship Model

Advantages of ER Model*:*

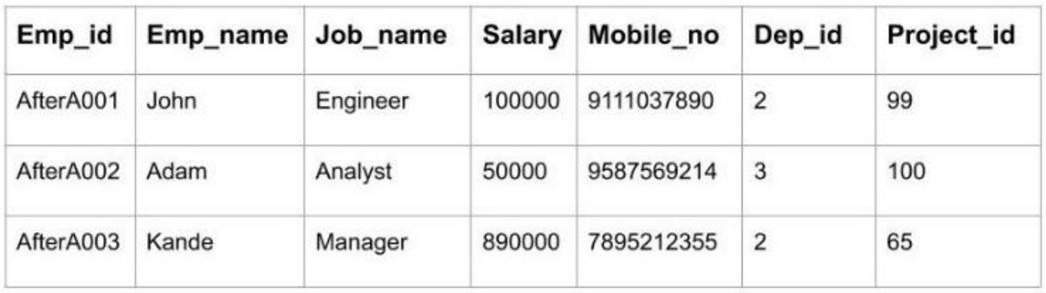
- **Simple**
- **Effective Communication Tool**
- **Easy Conversion to any Model**

Disadvantages of ER Model

- **No industry standard for notation**
- **Hidden information**

4.Relational Model:

Relational Model is the most widely used model. In this model, the data is maintained in the form of a two-dimensional table. All the information is stored in the form of row and columns. The basic structure of a relational model is tables. So, the tables are also called relations in the relational model.



Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

Advantages of Relational Model:

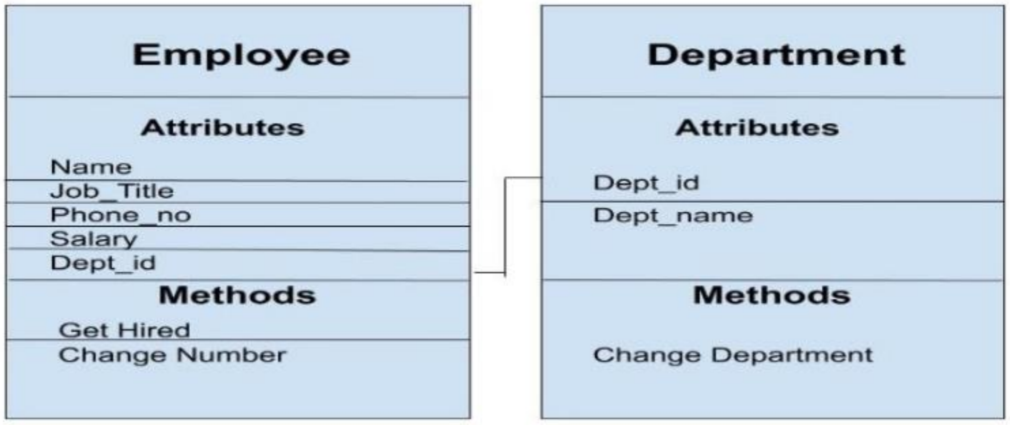
- **Simple**
- **Scalable**
- **Structural Independence**

Disadvantages of Relational Model:

- **Hardware Overheads**
- **Bad Design**

5.Object-Oriented Data Model

The real-world problems are more closely represented through the object-oriented data model. In this model, both the data and relationship are present in a single structure known as an object. In this model, two are more objects are connected through links.



Object_Oriented_Model

6. Object-Relational Model

It is a combination of both the relational model and the object-oriented model. This model was built to fill the gap between object-oriented model and the relational model. We can have many advanced features like we can make complex data types according to our requirements using the existing data types. The problem with this model is that this can get complex and difficult to handle. So, proper understanding of this model is required.

7. Flat Data Model

It is a simple model in which the database is represented as a table consisting of rows and columns. To access any data, the computer has to read the entire table. This makes the modes slow and inefficient.

8. Semi-Structured Model

Semi-structured model is an evolved form of the relational model. We cannot differentiate between data and schema in this model.

9. Associative Data Model

Associative Data Model is a model in which the data is divided into two parts. Everything which has independent existence is called as an entity and the relationship among these entities are called association. The data divided into two parts are called items and links.

Item: Items contain the name and the identifier(some numeric value).

Links: Links contain the identifier, source, verb and subject.

10.Context Data Model

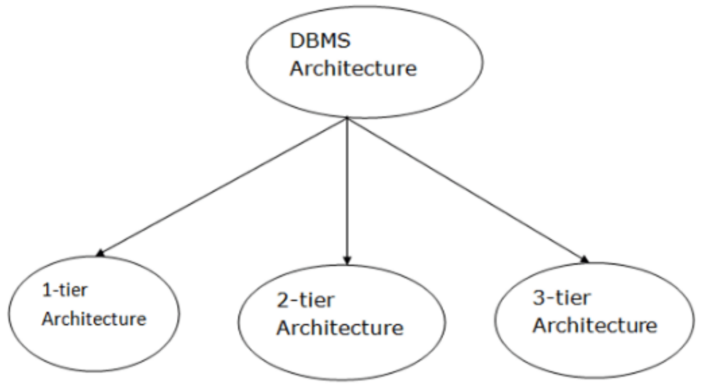
Context Data Model is a collection of several models. This consists of models like network model, relational models etc. Using this model we can do various types of tasks which are not possible using any model alone.

2. Utilize the knowledge of two-tier architecture and three-tier architecture to construct a well labeled diagram, illustrating the distinctive features and components of each.

DBMS Architecture:

- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture:



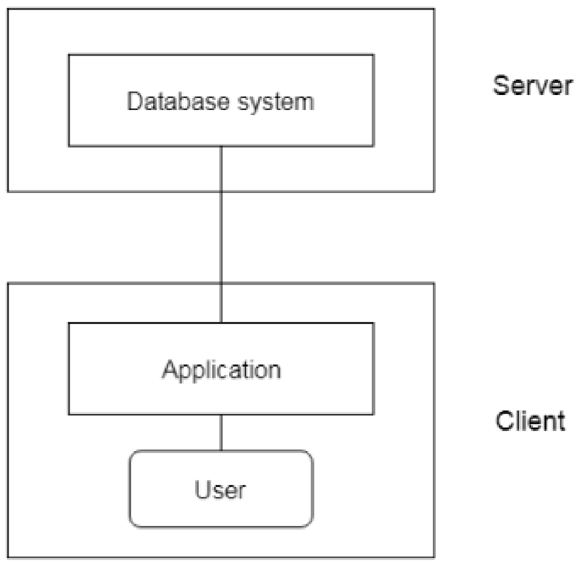
Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.
- A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.
- Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.



3-Tier Architecture

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

A 3-tier architecture has the following layers:

- ❖ Presentation layer (your PC, Tablet, Mobile, etc.)
- ❖ Application layer (server)
- ❖ Database Server

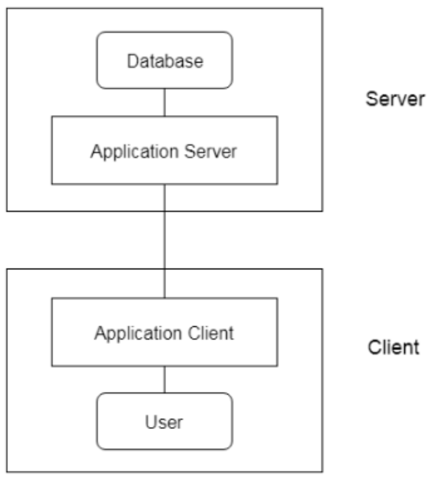
Database (Data) Tier − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

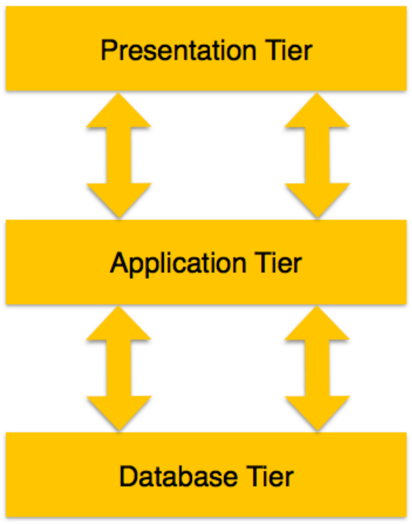
Application (Middle) Tier − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

User (Presentation) Tier − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

The goal of Three Tier client-server architecture is:

- ❖ To separate the user applications and physical database
- ❖ To support DBMS characteristics
- ❖ Program-data independence
- ❖ Supporting multiple views of the data





3. Compose a detailed explanation, accompanied by a clear diagram, outlining all the components of a DBMS.

Database System Architecture

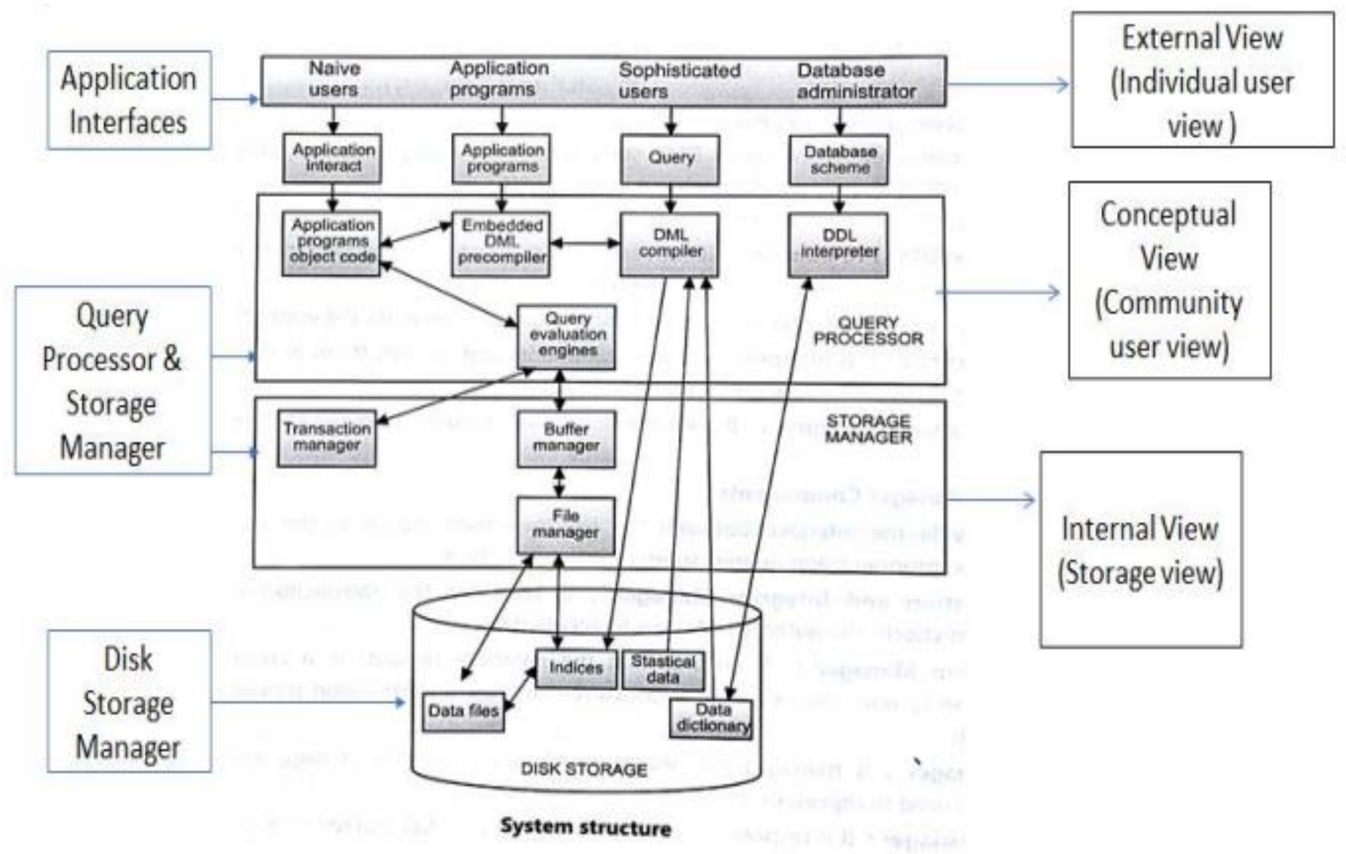
A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager, Database users and administrators and the query processor components.

The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data. A gigabyte is approximately 1000 megabytes (actually 1024) (1 billion bytes), and a terabyte is 1 million megabytes (1 trillion bytes). Since the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed. Since the movement of data to and from disk is slow relative to the speed of the central processing unit, it is imperative that the database system structure the data so as to minimize the need to move data between disk and main memory.

The query processor is important because it helps the database system to simplify and facilitate access to data. The query processor allows database users to obtain good performance while being able to work at the view level and not be burdened with understanding the physical-level details of the implementation of the system. It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.

Components of a database system can be divided into 3 units:

- ❖ Storage Manager
- ❖ Query Processor
- ❖ Database users and administrators.



System structure

1. Storage Manager:

It is a program module that provides an interface between the database and application program. The storage manager is responsible for storing, retrieving and updating the data in the database. A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system

1.1 Storage Manager Components:

Authorization and Integrity manager – Which tests for the satisfaction of integrity constraints and checks the authority of users to access data.

Transaction Manager: A transaction is a collection of operations that performs a single logical function in a database application. Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g. power failures and operating system crashes) and transaction failures. Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

File Manager -Which manages the allocation of storage space on disk and data structures used to store those information's

Buffer Manager – which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

2. Query Processor:

It interprets the requests (queries) received from end user via an application program into instructions. It also executes the user request which is received from the DML compiler.
Query Processor contains the following components :

DML Compiler :
It processes the DML statements into low level instruction (machine language), so that they can be executed.

DDL Interpreter :
It processes the DDL statements into a set of table containing meta data (data about data).

Embedded DML Pre-compiler:
It processes DML statements embedded in an application program into procedural calls.

Query Optimizer:
It executes the instruction generated by DML Compiler.

3a. Database Users

1) **Application programmers** – are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. **Rapid application development (RAD)** tools are tools that en-able an application programmer to construct forms and reports with minimal programming effort.
2) **Sophisticated users** – interact with the system without writing programs. In-stead, they form their requests either using a database query language or by using tools such as data analysis software. Analysts who submit queries to explore data in the database fall in this category.

3) **Specialized user** – are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledge-base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems. Chapter 22covers several of these applications.
4) **Naïve users** – are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. For example, a clerk in the university who needs to add a new instructor to department A invokes a program called new hire. This program asks the clerk for the name of the new instructor, her new ID, the name of the department(that is, A), and the salary.

3b.Database Administrator

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 ofdata in case of disasters such as flooding.

✓ Ensuring that enough free disk space is available for normal operations, and upgrading disk spaceas required.

✓ Monitoring jobs running on the database and ensuring that performance is not degraded by veryexpensive tasks submitted by some users.

4.. Disk Storage:

- **Data files**, which store the database itself.
- **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
- **Indices**, which can provide fast access to data items. A database index provides pointers to those data items that hold a particular value. For example, we could use an index to find the instructor record with a particular ID, or all instructor records with a particular name. Hashing is an alternative to indexing that is faster in some but not all cases.

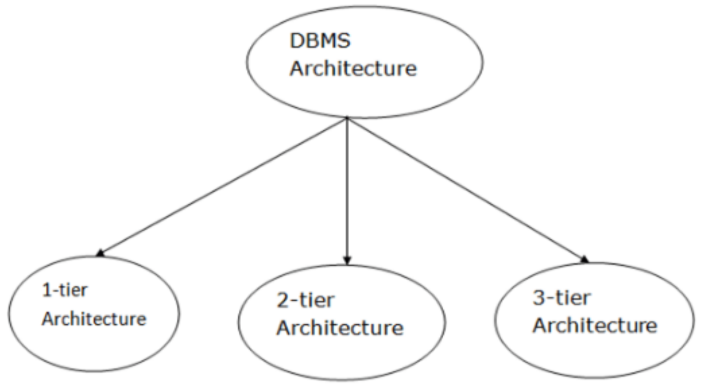
DBMS Architecture:

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

❖ The client/server architecture consists of many PCs and a workstation which are connected via the network.

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

Types of DBMS Architecture:



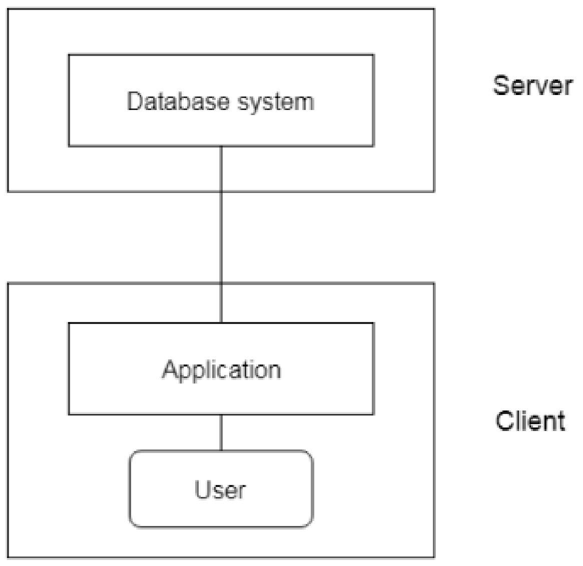
Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.
- A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.
- Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.



3-Tier Architecture

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

A 3-tier architecture has the following layers:

- ❖ Presentation layer (your PC, Tablet, Mobile, etc.)
- ❖ Application layer (server)
- ❖ Database Server

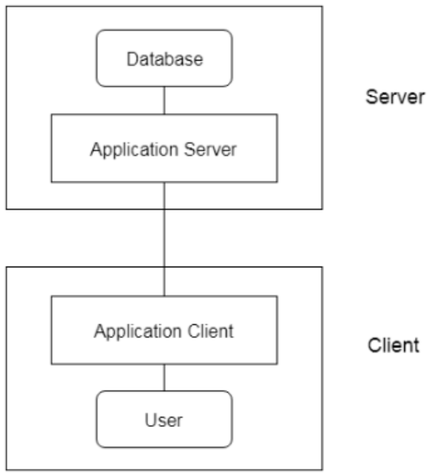
Database (Data) Tier − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

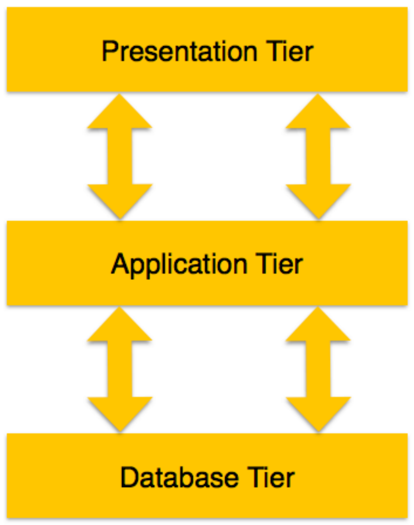
Application (Middle) Tier − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

User (Presentation) Tier − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

The goal of Three Tier client-server architecture is:

- ❖ To separate the user applications and physical database
- ❖ To support DBMS characteristics
- ❖ Program-data independence
- ❖ Supporting multiple views of the data





4. Apply the understanding of basic relational algebra operations by providing examples that illustrate the practical application of each operation.

Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

- **Select**
- **Project**
- **Union**
- **Set different**
- **Cartesian product**
- **Rename**

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 =, ≠, ≥, <, >, ≤.
For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

```
σ BRANCH_NAME="perryride" (LOAN)
```

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

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	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

```
∏ NAME, CITY (CUSTOMER)
```

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.
- Notation: R ∪ S

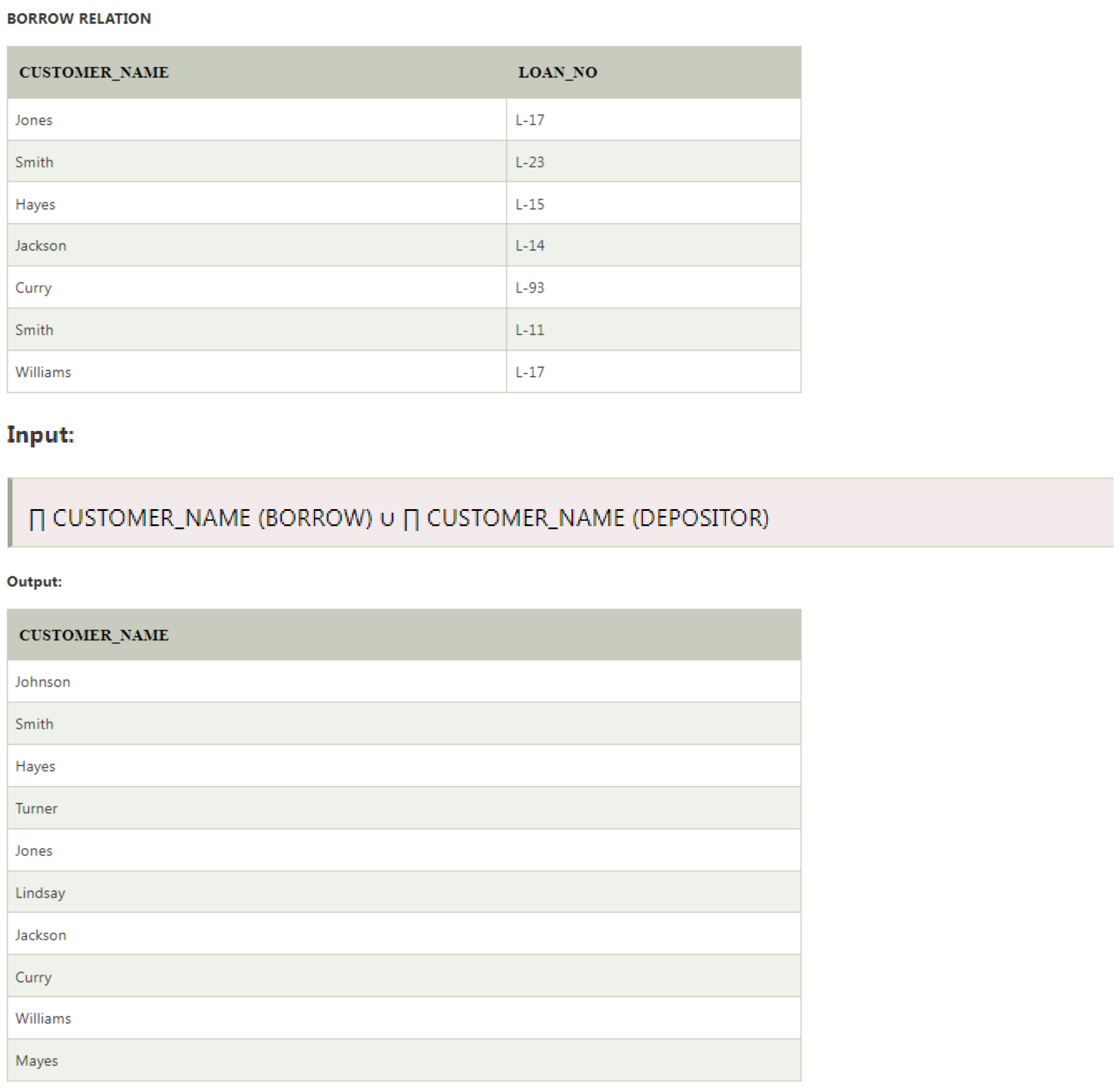
A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284



BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:

```
∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)
```

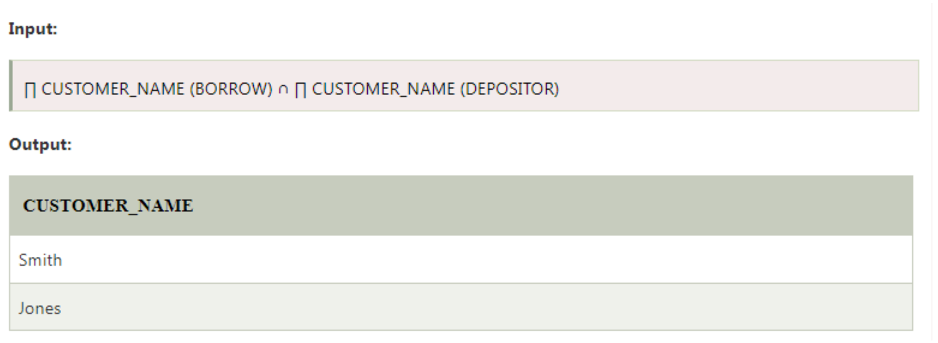
Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

  Notation: R ∩ S



Input:

```
∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)
```

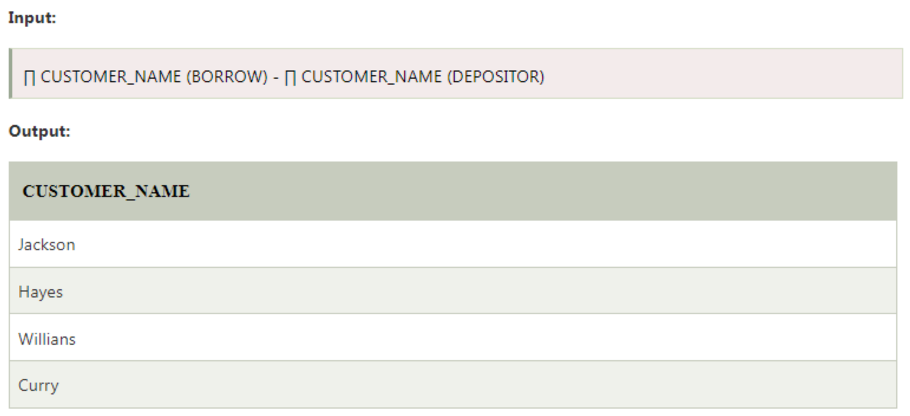
Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

Notation: R - S
Example: Using the above DEPOSITOR table and BORROW table



Input:

```
∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)
```

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

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

Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

```
EMPLOYEE X DEPARTMENT
```

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. 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)
```

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:

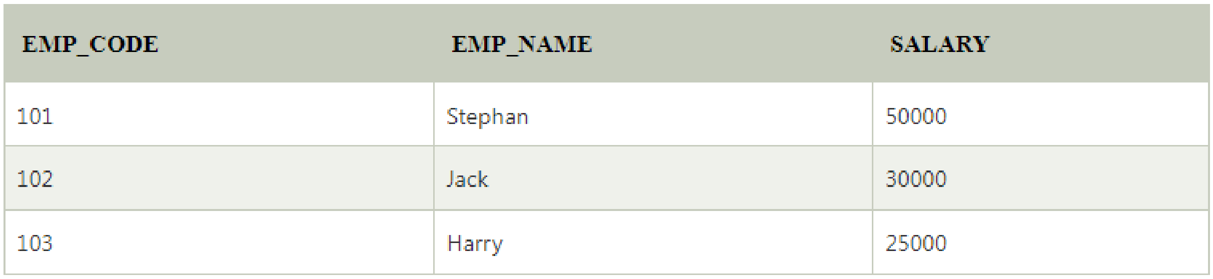
EMPLOYEE

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

SALARY

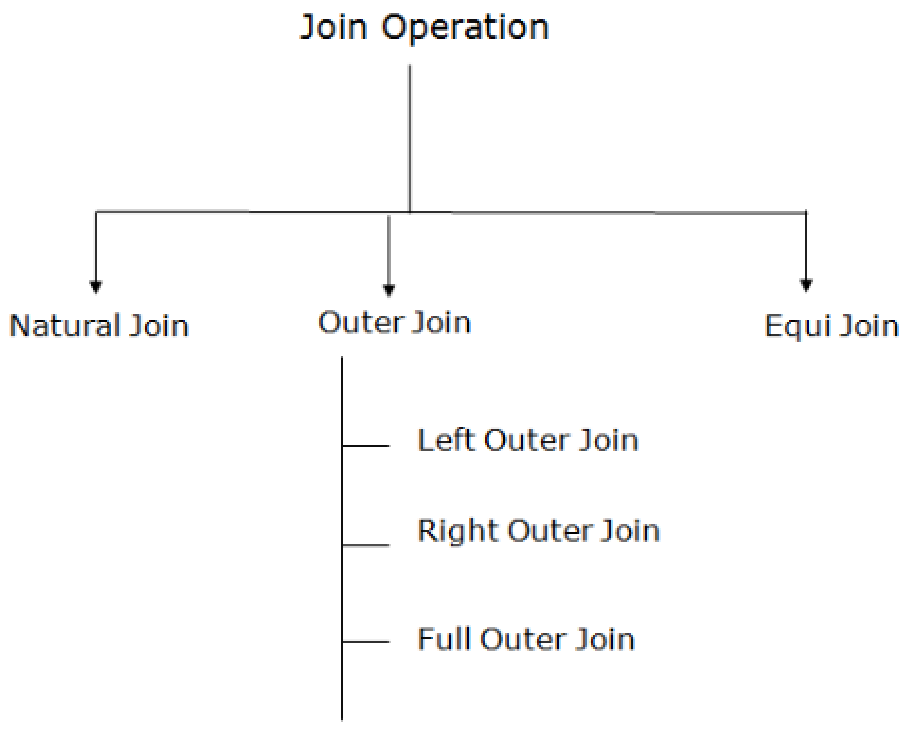
EMP_CODE	SALARY
101	50000
102	30000
103	25000

Operation: (EMPLOYEE ⋈ SALARY)



EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

Types of Join operations:



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:

Input:

1. ∏EMP_NAME, SALARY (EMPLOYEE ⋈ SALARY)

Output:

EMP_NAME	SALARY
Stephan	50000
Jack	30000

Harry	25000

2. Outer Join:

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

Example:

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

Input: (EMPLOYEE ⋈ FACT_WORKERS)

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru nagar	Hyderabad	TCS	50000

An outer join is basically of three types:

a. Left outer join
b. Right outer join

c. 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

Input:

EMPLOYEE ⟕ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

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

Input:

EMPLOYEE ⟖ FACT_WORKERS

Output:

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai

Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

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

Input:

EMPLOYEE ⟗ FACT_WORKERS

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

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 RELATION

CLASS_ID	NAME
1	John
2	Harry

3	Jackson

PRODUCT

PRODUCT_ID	CITY
1	Delhi
2	Mumbai
3	Noida

Input:

CUSTOMER ⋈ PRODUCT

Output:

CLASS_ID	NAME	PRODUCT_ID	CITY
1	John	1	Delhi
2	Harry	2	Mumbai
3	Harry	3	Noida

5. Consider a relation Employee (Employee_name, Company_name, Salary). Analyze and Write SQL for the following

(i) the total salary of each company

(ii) Find the employee name who is getting highest salary

(iii)Find the company name which has lowest average salary

(iv) Find the employee name whose salary is higher than average salary of TCS.

(i) To find the total salary of each company:

```
SELECT Company_name, SUM(Salary) AS Total_salary
FROM Employee
GROUP BY Company_name;
```

(ii) To find the employee name who is getting the highest salary:

```
SELECT Employee_name

FROM Employee

WHERE Salary = (SELECT MAX(Salary) FROM Employee);
```

(iii) To find the company name which has the lowest average salary:

```
SELECT Company_name
FROM (
   SELECT Company_name, AVG(Salary) AS Avg_salary
   FROM Employee
   GROUP BY Company_name
   ORDER BY Avg_salary ASC
   LIMIT 1
) AS Lowest_avg_salary_company;
```

(iv) To find the employee name whose salary is higher than the average salary of TCS:

```
SELECT Employee_name

FROM Employee

WHERE Salary > (SELECT AVG(Salary) FROM Employee WHERE Company_name = 'TCS');
```