UNIT 1

Database management system is software that is used to manage the database.

What is Database

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

For example: The college Database organizes the data about the admin, staff, students and faculty etc.

Using the database, you can easily retrieve, insert, and delete the information.

Database Management System

- Database management system is a software which is used to manage the database. For example: MySQL, Oracle , etc are a very popular commercial database which is used in different applications.
- DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

DBMS allows users the following tasks:

- **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
- **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
- **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes

Advantages of DBMS

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.

- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.
- **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

Database-System Applications

1 Enterprise Information:

- Sales
- Accountin
- Human resource
- Manufacturing
- Online retailers

2. Banking & Finance:

- Banking
- Credit card transactios.
- Finance.

3. Universities

4 Airlines

5. Tele-communications

DBMS Vs FILE

- 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.

- Searching data is easy in Dbms

Searching is difficult in File System

- Dbms is structured data

Files are unstructured data

- No data redundancy in Dbms

Data redundancy is there in file system

- Memory utilisation well in dbms

Memory utilisation poor in file system

- No data inconsistency in dbms

Inconsistency in file system.

- 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 contains a wide variety of sophisticated techniques to store and retrieve the data.

File system can't efficiently store and retrieve the data.

PURPOSE OF DATABASE SYSTEMS

The purpose of DBMS can be observed from the drawbacks of the prior system that existed before DBMS and i.e file-processing system.

Characteristics of DBMS

Data stored into Tables: Data is never directly stored into the database. Data is stored into tables, created inside the database.

• **Reduced Redundancy:** DBMS follows Normalisation which divides the data in such a way that repetition is minimum.

• **Data Consistency:** On Live data, i.e. data that is being continuosly updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.

• **Support Multiple user and Concurrent Access:** DBMS allows multiple users to work on it (update, insert, delete data) at the same time and still manages to maintain the data consistency

- **Multiple user and Concurrent Access:** DBMS allows multiple users to work on it (update, insert, delete data) at the same time and still manages to maintain the data consistency
- **Query Language:** DBMS provides users with a simple Query language, using which data can be easily retrieve, inserted, deleted and updated in a database

View of Data in DBMS

• Abstraction is one of the main features of database systems.

• Hiding irrelevant details from user and providing abstract view of data to users, helps in easy and efficient user-database interaction.

• the three level of DBMS architecture, The top level of that architecture is “view level”. The view level provides the “view of data” to the users and hides the irrelevant details such as data relationship, database schema, security etc from the user.

Data Abstraction in DBMS

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.

DIAGRAM IN NOTES:

We have three levels of abstraction:

1. **Physical level:** This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.
2. **Logical level:** This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.
3. **View level:** Highest level of data abstraction. This level describes the user interaction with database system.

Definition of schema:

Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

• The design of a database at physical level is called physical schema, how the data are stored

• Design of database at logical level is called logical schema, programmers and database administrators work at this level

• Design of database at view level is called view schema. This generally describes end user interaction with database systems

Definition of instance:

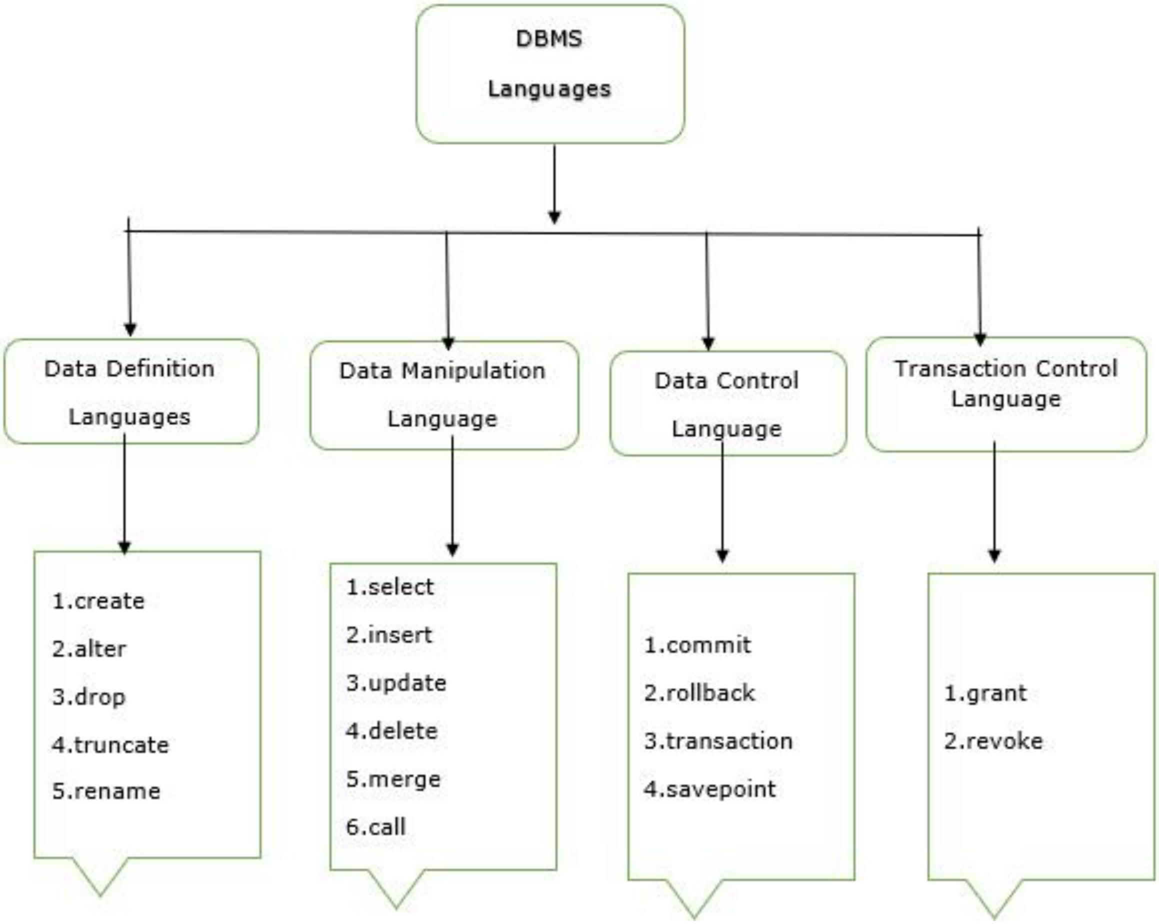
The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

Database languages:

It is used to read, update and store data in a database. There are several such languages that can be used for this purpose; one of them is SQL (Structured Query Language).

DIAGRAM IN NOTES

- DDL – Data Definition Language: (CREATE,DROP,ALTER,TRUNCATE,COMMENT,RENAME)
- DML – Data Manipulation Language: (INSERT, UPDATE,DELETE)
- DCL – Data Control Language: (GRANT,REVOKE) • TCL-Transaction Control Language: (COMMIT,ROLLBACK)



7 What are the types of languages a database system provides? Explain.

ANS**: Data Definition Language (DDL**) :

It is a language that allows the users to define data and their relationship to other types of data. It is mainly used to create files, databases, data dictionary and tables within databases.

It is also used to specify the structure of each table, set of associated values with each attribute, integrity constraints, security and authorization information for each table and physical storage structure of each table on disk.

Data Manipulation Language (DML)

It is a language that provides a set of operations to support the basic data manipulation operations on the data held in the databases

It allows users to insert, update, delete and retrieve data from the database.

Data Control Language (DCL):

DCL statements control access the stored data using statements such as GRANT and REVOKE. A privilege can either be granted to a User with the help of GRANT statement. The privileges assigned can be SELECT, ALTER, DELETE, EXECUTE, INSERT, INDEX etc. In addition to granting of privileges, you can also revoke (taken back) it by using REVOKE command.

In practice, the data definition and data manipulation languages are not two separate languages. Instead they simply form parts of a single database language such as Structured Query Language (SQL). SQL represents combination of DDL and DML, as well as statements for constraints specification and schema evaluation.

TCL (Transaction Control Language) :

Transaction Control Language commands are used to manage transactions in the database. These are used to manage the changes made by DML-statements. It also allows statements to be grouped together into logical transactions.

Examples of TCL commands – COMMIT: Commit command is used to permanently save any transaction into the database.

ROLLBACK: This command restores the database to last committed state.

It is also used with savepoint command to jump to a savepoint

in a transaction.

SAVEPOINT: Savepoint command is used to temporarily save a transaction so

that you can rollback to that point whenever necessary.

Q. What is a data model? Explain in detail about different data models used in database management systems?

Ans: **Data Model** – Model is an abstraction process that hides irrelevant details while highlighting details relevant to the applications at hand. Similarly, a data model is a collection of concepts that can be used to describe structure of a database and provides the necessary means to achieve this abstraction. Structure of database means the data types, relationships, and constraints that should hold for the data.

In general a data model consists of two elements:

• A mathematical notation for expressing data and relationships.

• Operations on the data that serve to express queries and other manipulations of the data.

Data Models used in DBMSs:

• **Hierarchical Model -** It was developed to model many types of hierarchical organizations that exist in the real world. It uses tree structures to represent relationship among records. In hierarchical model, “no dependent record can occur without its parent record occurrence” and no “dependent record occurrence may be connected to more than one parent record occurrence.”

• **Network Model** - It was formalized in the late 1960s by the Database Task Group of the Conference on Data System Language (DBTG/CODASYL). It uses two different data structures to represent the database entities and relationships between the entities, namely record type and set type. In the network model, the relationships as well as the navigation through the database are predefined at database creation time.

• **Relational Model** - The relational model was first introduced by E.F. Codd of the IBM Research in 1970. The model uses the concept of a mathematical relation (like a table of values) as its basic building block, and has its theoretical basis in set theory and first-order predicate logic. The relational model represents the database as a collection of relations.

- **Object Oriented Model** – This model is based on the object-oriented programming language paradigm. It includes the features of OOP like inheritance, object-identity, encapsulation, etc. It also supports a rich type system, including structured and collection types.

- **Object Relational Model** – This model combines the features of both relational model and object oriented model. • It extends the traditional relational model with a variety of features such as structured and collection types.

Short Answer Questions

1 **Define the terms instance and schema.**

Instance: The data stored in database at a particular moment of time is called instance of database.

Schema: Design of a database is called the schema. Schema is of three types: Physical schema,logical schema and view schema.

2 **List and explain five reference options in a query.**

Casacade:If a row from the parent table is deleted or updated then the corresponding child is also effected.

Set null: If a row from parent table is deleted or updated then the values of foreign key column in child table is set to null

Restrict:IC restricting update and delete in parent table when it has matching rows in child table.No action: it is similar to restrict.

3 What is storage Manager?

A storage Manager is a program module which is responsible for storing , retrieving and updating data in the database.

There are Four components of storage Manager , They are:

1. Authorization and Integrity manger

2. Transaction Manager

3. File Manager

4. Buffer Manager

4 **Define the terms Entity set , Relationship set?**

Ans:

Entity Set:

set is a set of entities of same type . An entity set may be of two types :

1. Strong Entity Set

2. Weak Entity Set

RelationshipSet :

A relationship set is a set of relationships of same type . A relationship set may be a unary relationship set or binary relationship set or ternary relationship set or n-ary relationship set.

5 Define weak and strong entity sets?

Strong Entity:

Strong entity is not dependent of any other entity in schema. Strong entity always has primary key. Strong entity is represented by single rectangle. Two strong entity’s relationship is represented by single diamond.

Various strong entities together makes the strong entity set.

Weak Entity:

Weak entity is dependent on strong entity to ensure the existence of weak entity. Like strong entity weak entity does not have any primary key, It has partial discriminator key.

Weak entity is represented by double rectangle.

The relation between one strong and one weak entity is represented by double diamond.

6 Define NULL values?

SQL supports a special value known as NULL which is used to represent the values of attributes that may be unknown or not apply to a tuple. .

- It is important to understand that a NULL value is different from zero value.

- A NULL value is used to represent a missing value, but that it usually has one of three different interpretations:

- Value unknown (value exists but is not known)

- Value not available (exists but is purposely withheld)

- Attribute not applicable (undefined for this tuple)

It is often not possible to determine which of the meanings is intended. Hence, SQL does not distinguish between the different meanings of NULL

VIEW ON TABLES , WITH SUITABLE QUERY?

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

CREATING VIEWS

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

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

Creating View from a single table

In this example we will create a View named DetailsView from the table StudentDetails.

Query:

```
CREATE VIEW Details View AS
SELECT NAME, ADDRESS
FROM Student Details
WHERE S_ID < 5;
```

Creating View from multiple tables:

```
CREATE VIEW Marks View AS
SELECT StudentDetails.NAME, StudentDetails. ADDRESS, StudentMarks .MARKS
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME
```

Query for creating and updating a view

Creating a view:

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM StudentDetails
WHERE S_ID < 5;
```

Updating a view:

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1,coulmn2,..
FROM table_name
WHERE condition;
```

logical independencies:

1. Ability to change the logical (conceptual) schema without changing the external schema(user view)
2. Modification is necessary whenever the logical structure of the database is altered.
3. It means we will be able to change the logical schema of the database without effecting the view level.
4. It is more difficult to achieve.
5. For example: Consider two users A & B, both are selecting the fields “employee_number”. If user B adds new

column “salary” to the same table, it will

not affect the user view level.

Physical Data Independence :

1. Ability to change the physical schema

without changing the logical schema.

2. Modification are rarely done, as it is done to

improve the performance of the database.

3. It means we will be able change the physical

storage without affecting the logical level or

view level.

4. It is relatively easy to achieve.

5. For example: Changing the storage drive of

the data base form “C” drive to “D” drive

will not affect the conceptual and view level

as the new changes are absorbed by the

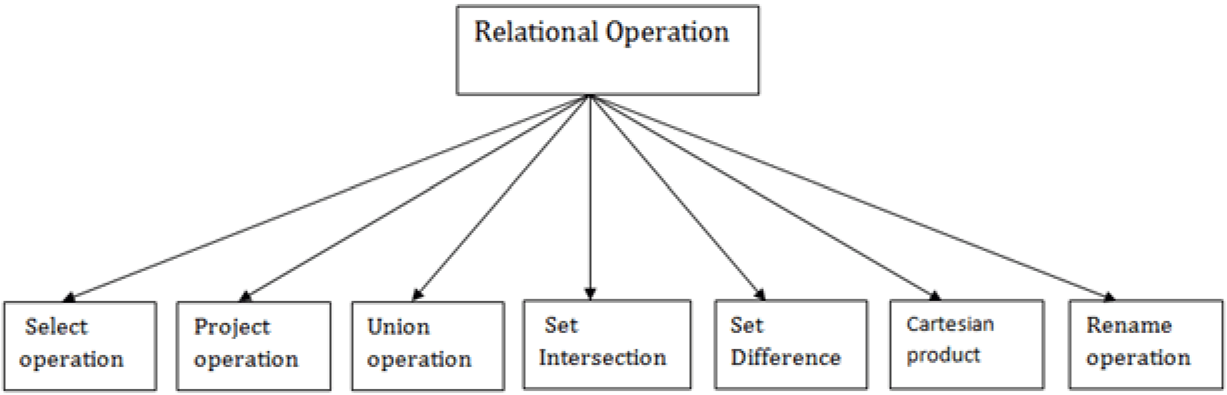
mapping technique.

Entity Relationship (ER) for railway reservation system

Relational Algebra

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.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1. 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:

1. 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 ∏.

1. 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 ∪.

1. 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::

1. ∏ 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 ∩.

1. Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

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 (-).

1. Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ 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.

1. 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:

1. 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.

1. 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

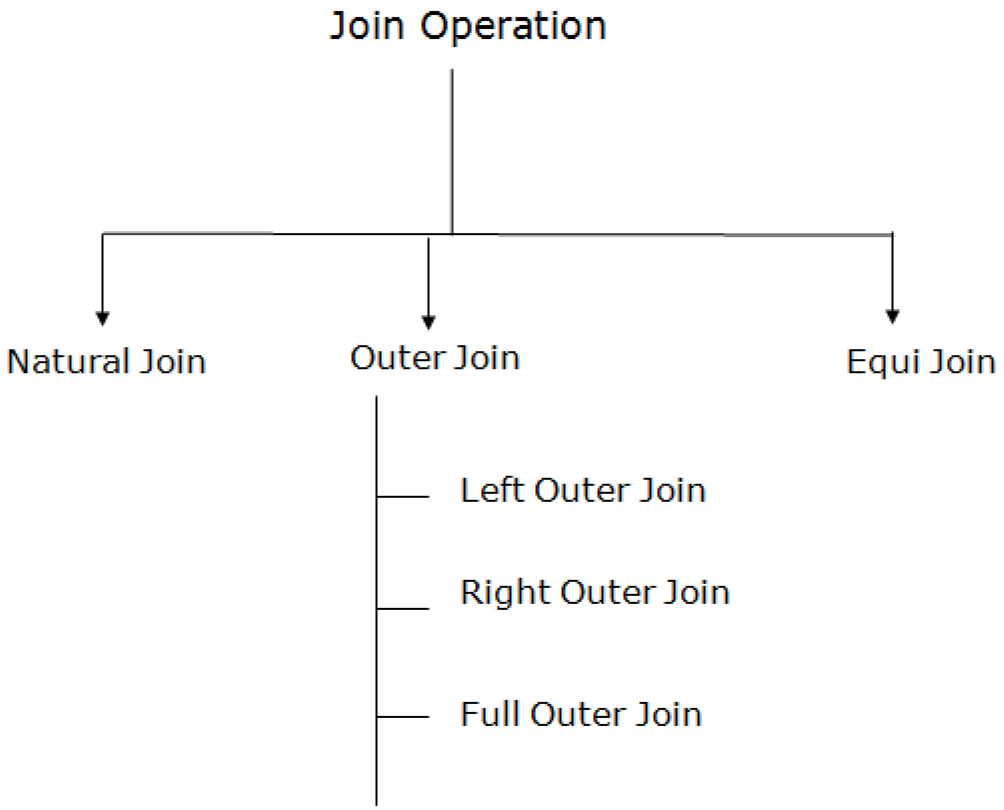
SALARY

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:

1. (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

- LEFT OUTER JOIN - keep data from the left-hand table

- RIGHT OUTER JOIN - keep data from the right-hand table

- FULL OUTER JOIN - keep data from both tables

Left outer join:

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

Right outer join:

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

Full outer join:

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