Structured Query Language

Module Name: **Relational Database Management System**

Module Code: **CS 1211**

Instructors: **Jamila L.**



Structured Query Language

Originally developed in 1974 by D. D. Chamberlin and others at IBM's San Jose Research Laboratory, SQL was finally released to the public as an integral part of IBM's database products in 1982-1983.

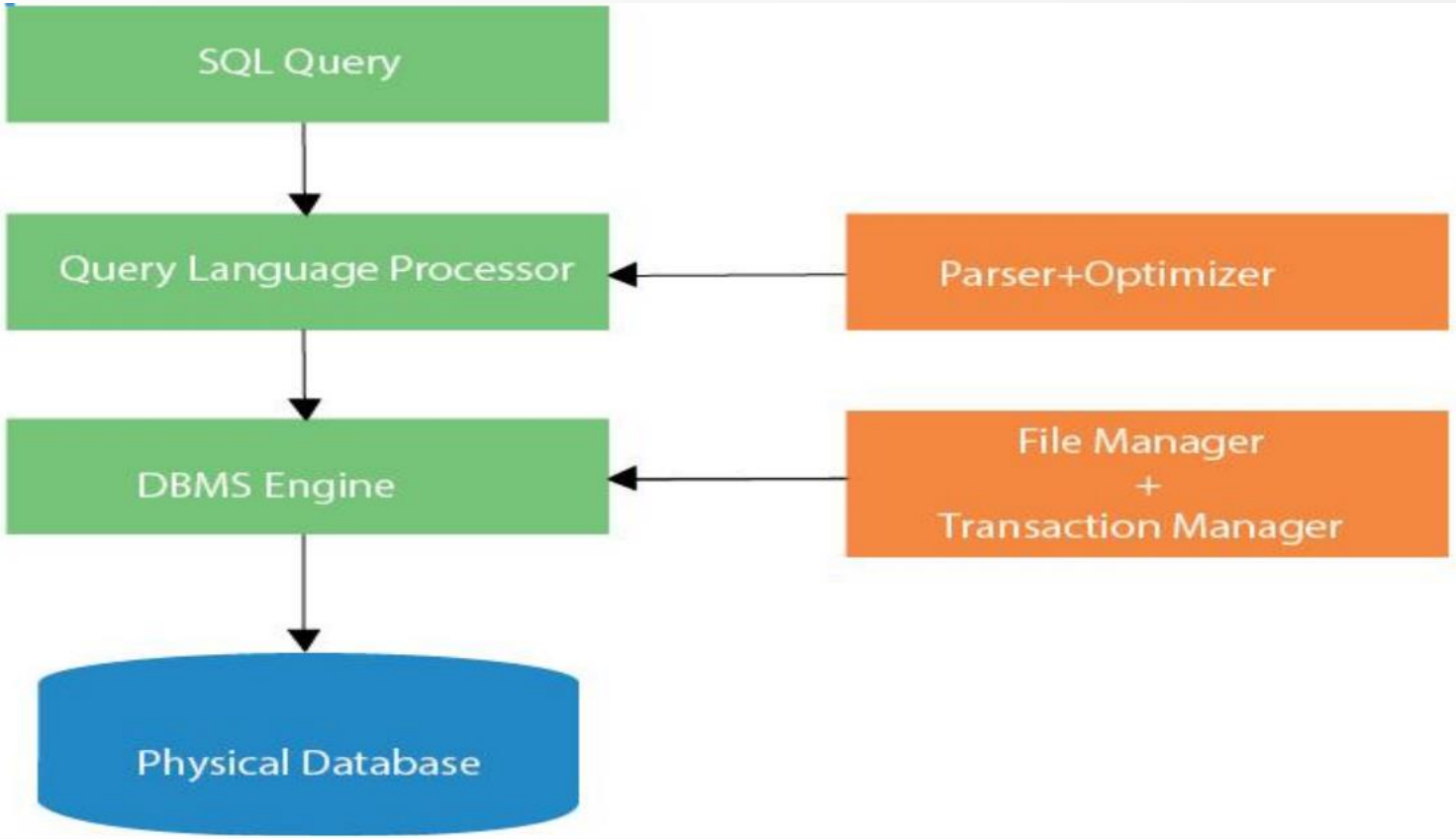
- SQL was adopted as an official standard by the American National Standards Institute (ANSI) in 1986, and in 1987 by the International Organization for Standardization (ISO). It is currently available in over 100 commercial products.

What is SQL?

- SQL is a standard language for creating, accessing and manipulating databases.
- A SQL statement is a collection of clauses, keywords, and parameters that perform a particular function.

However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

SQL Process


SQL Query
Query Language Processor
Parser+Optimizer
DBMS Engine
File Manager
+
Transaction Manager
Physical Database

What is Advantages

- High speed
- No coding needed
- Well defined standards
- Portability
- Interactive language
- Multiple data view

SQL Commands

SQL commands are instructions used to communicate with the database, used to perform specific tasks, functions, and queries of data.

SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands

i. Data Definition Language (DDL)
ii. Data Manipulation Language (DML)
iii. Data Query Language (DQL)
iv. Data Control Language (DCL)
v. Transaction Control Language (TCL)

SQL Commands: DDL

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

• All the command of DDL are auto-committed, that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

SQL Commands: DDL

CREATE:

CREATE It is used to create a new table in the database.

Syntax:

CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Example:

CREATE TABLE employee_tbl(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

SQL Commands: DDL

DROP:

Drop: It is used to delete both the structure and record stored in the table.

Syntax:

DROP TABLE table_name;

Example:

DROP TABLE employee_tbl;

SQL Commands: DDL

ALTER:

It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

ALTER TABLE table_name ADD column_name COLUMN-definition;

ALTER TABLE MODIFY(COLUMN definition....);

Example:

ALTER TABLE studentData_tbl ADD(address VARCHAR2(20));

ALTER TABLE studentData_tbl MODIFY (name VARCHAR2(20));

SQL Commands: DDL

TRUNCATE:

It is used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table_name;

Example:

TRUNCATE TABLE employee_tbl;

SQL Commands: DML

Data Manipulation Language (DML) commands are used to modify the database. It is responsible for all form of CHANGES in the database.

- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

SQL Commands: DML

INSERT:

The INSERT statement is a SQL query used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE_NAME (col1, col2, col3,.... col N)

VALUES (value1, value2, value3, ... valueN);

OR

INSERT INTO TABLE_NAME VALUES (value1, value2, value3, ... valueN);

Example:

INSERT INTO books_tbl (Author, Subject) VALUES ("Sonoo", "DBMS");

SQL Commands: DML

UPDATE:

This command is used to update or modify the value of acolumn in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION];

Example:

UPDATE students_tbl SET User_Name ='Sonoo' WHERE Student_Id = '3';

SQL Commands: DML

DELETE:

The DELETE statement is used to delete existing records in a table.

Syntax:

DELETE FROM *table_name* WHERE *condition*;

Example:

DELETE FROM Customers WHERE CustomerID='ca0007';

SQL Commands: DQL

Data Query Language (DQL) is used to fetch the data from the database. It uses only one command:

SELECT:

It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

SELECT expressions FROM TABLES WHERE conditions;

SELECT column1, column2, ... FROM table_name;

Example:

SELECT * FROM Customers_tbl;

SELECT employee_name FROM employee_tbl WHERE age > 20;

SELECT CustomerName, City FROM Customers_tbl;

SQL Commands: DCL

Data Control Language (DCL) commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

SQL Commands: DCL

GRANT:

It is used to give user access privileges to a database.

Example:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

REVOKE:

It is used to take back permissions from the user.

Example:

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

SQL Commands: TCL

Transaction Control Language (TCL) commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

SQL Commands: TCL

Commit:

Commit command is used to save all the transactions to the database.

Syntex:

COMMIT;

Example:

DELETE FROM customers WHERE age = 25;

COMMIT;

SQL Commands: TCL

Rollback:

Rollback command is used to undo transactions that have not already been saved to the database.

Syntex:

ROLLBACK;

Example:

DELETE FROM customers WHERE age = 25;

ROLLBACK;

SQL Commands: TCL

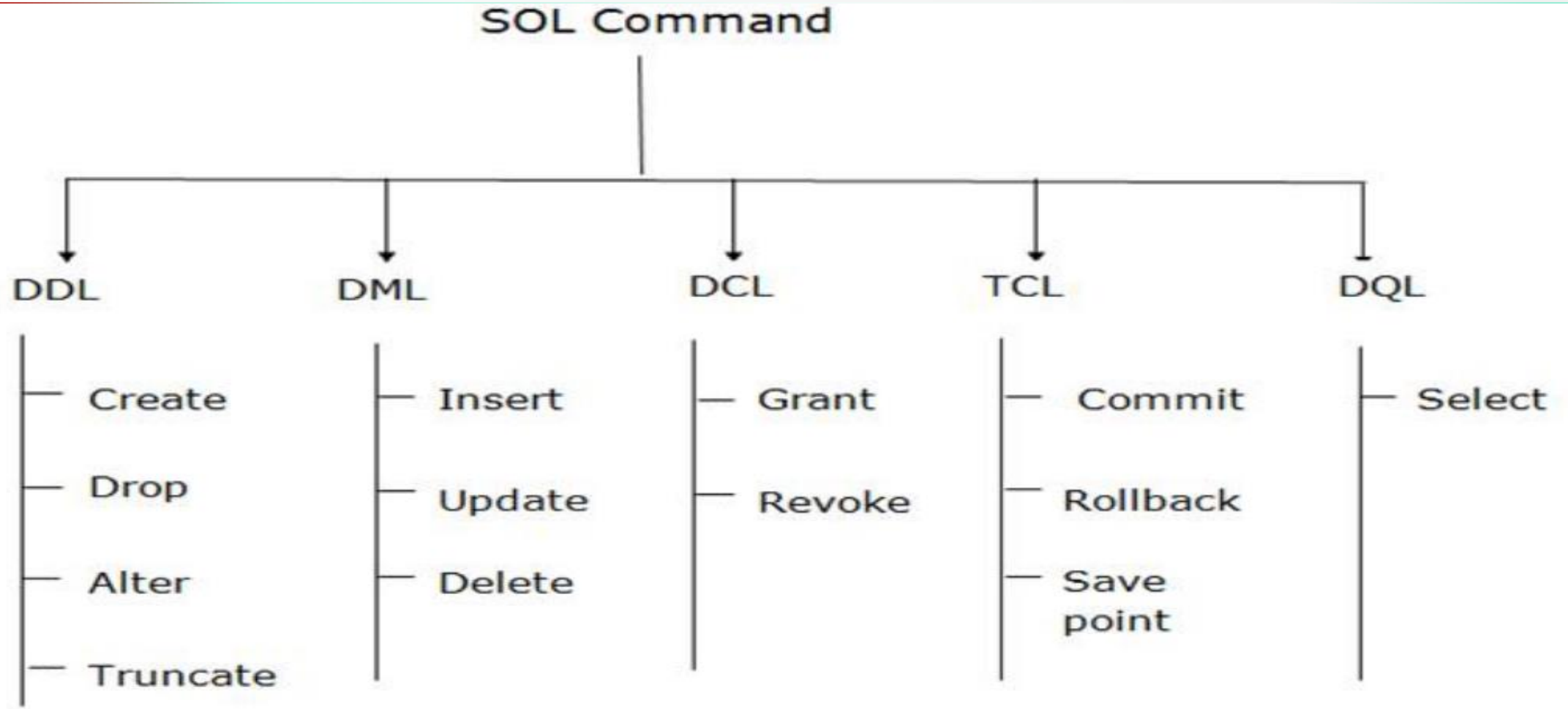
SAVEPOINT:

It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntex:

SAVEPOINT SAVEPOINT_NAME;

SQL Commands



Formal query Languages

Formal Query Languages

Formal query languages Formal query languages are mathematical frameworks used to manipulate and retrieve data from relational databases.

- Unlike general programming languages, they are specialized for efficient data access rather than complex calculations

Core Categories:

Formal relational query languages are broadly classified based on how they process data:

1. **Procedural (Relational Algebra):** The user defines a series of operations to compute the result. It describes the "how" of data retrieval.
2. **Non-procedural (Relational Calculus):** The user describes what information is desired without specifying the steps to find it. It is a declarative approach based on predicate logic.

Relational Algebra

Relational Algebra is a formal language used to query and manipulate relational databases, consisting of a set of operations like selection, projection, union, and join.

- It provides a mathematical framework for querying databases, ensuring efficient data retrieval and manipulation.
- Relational algebra serves as the mathematical foundation for SQL query.
- SQL queries are based on relational algebra operations, enabling users to retrieve data effectively.

Basic Operators in Relational Algebra

Relational algebra consists of various basic operators that help us to fetch and manipulate data from relational tables in the database to perform certain operations on relational data.

- Basic operators are:
  i. selection (σ)
  ii. projection (π)
  iii. union (U)
  iv. set difference (−)
  v. Cartesian product (×)
  vi. rename (ρ).

1. Selection(σ)

The Selection Operation is basically used to filter out rows from a given table based on certain given condition. It basically allows us to retrieve only those rows that match the condition as per condition passed during SQL Query.

Example: If we have a relation R with attributes A, B, and C, and we want to select tuples where C > 3, we write:

A	B	C
1	2	4
2	2	3
3	2	3
4	3	4

$\sigma_{(c>3)}(R)$ will select the tuples which have c more than 3.

Output:

A	B	C
1	2	4
4	3	4

2. Projection(π)

While Selection operation works on rows, similarly projection operation of relational algebra works on columns. It basically allows us to pick specific columns from a given relational table based on the given condition and ignoring all the other remaining columns.

Example:

Suppose we want columns B and C from Relation R.

$\pi_{(B,C)}(R)$ will show following columns.

By Default, projection operation removes duplicate values.

Output:

B	C
2	4
2	3
3	4

3. Union(U)

The Union Operator is basically used to combine the results of two queries into a single result.

The only condition is that both queries must return same number of columns with same data types. Union operation in relational algebra is the same as union operation in set theory.

Example: Consider the following table of Students having different optional subjects in their course.

FRENCH

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

GERMAN

Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25

3. Union(U) cont...

If FRENCH and GERMAN relations represent student names in two subjects, we can combine their student names as follows:

$\pi_{(Student_Name)}(FRENCH) \cup \pi_{(Student_Name)}(GERMAN)$

Output:

Student_Name
Ram
Mohan
Vivek
Geeta
Shyam
Rohan

The only constraint in the union of two relations is that both relations must have the same set of Attributes.

4. Set Difference(-)

Set difference basically provides the rows that are present in one table, but not in another tables. Set Difference in relational algebra is the same set difference operation as in set theory.

Example:

To find students enrolled only in FRENCH but not in GERMAN, we write:

$$\pi_{(Student_Name)}(FRENCH) - \pi_{(Student_Name)}(GERMAN)$$

The only constraint in the Set Difference between two relations is that both relations must have the same set of Attributes.

Output:

Student_Name
Ram
Mohan

5. Rename(ρ)

Rename operator basically allows to give a temporary name to a specific relational table or to its columns.

- It is very useful when we want to avoid ambiguity, especially in complex Queries. Rename is a unary operation used for renaming attributes of a relation.

Example: We can rename an attribute B in relation R to D

A	B	C
1	2	4
2	2	3
3	2	3
4	3	4

$\rho_{(D/B)}R$ will rename the attribute 'B' of the relation by "D".

Output:

A	D	C
1	2	4
2	2	3
3	2	3
4	3	4

6. Cartesian Product(^{X})

It Combines every row of one table with every row of another table, producing all the possible combination. It's mostly used as a precursor to more complex operation like joins. Let's say A and B, so the cross product between A ^{X} B will result in all the attributes of A followed by each attribute of B. Each record of A will pair with every record of B.

Relation A:

Name	Age	Sex
Ram	14	M
Sona	15	F
Kim	20	M

Relation B:

ID	Course
1	DS
2	DBMS

Output:

Name	Age	Sex	ID	Course
Ram	14	M	1	DS
Ram	14	M	2	DBMS
Sona	15	F	1	DS
Sona	15	F	2	DBMS
Kim	20	M	1	DS
Kim	20	M	2	DBMS

If A has 'n' tuples and B has 'm' tuples then A ^{X} B will have 'n*m' tuples.

Derived Operators in Relational Algebra

Derived operators are built using basic operators and include operations like join, intersection, and division.

These operators help perform more complex queries by combining basic operations to meet specific data retrieval needs.

Those operators are:

1. Join Operators:
    - Inner Join
    - Outer Join
2. Set Intersection(∩)
3. Division (÷)

Join Operators

Join is an operation in DBMS(Database Management System) that combines the rows of two or more tables based on related columns between them. The main purpose of join is to retrieve the data from multiple tables in other words Join is used to perform multi-table queries. It is denoted by ⋈

Join Combine data from two or more relations based on a related attribute, allowing for more complex queries and data retrieval.

Syntax:

$$R3 \leftarrow (R1) \bowtie_{\text{join_condition}} (R2)$$

where R1 and R2 are two relations to be joined and R3 is a relation that will hold the result of the join operation.

Example:

$$Temp \leftarrow (student) \bowtie_{S.roll=E.roll} (Exam)$$

Where S and E are aliases of the student and exam respectively.

Join Operators Example

Both these tables are connected by one common key (column) i.e. ROLL_NO.
We can perform a JOIN operation using the given relational algebra:

Student ⋈ Student_course

Table: Students

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	XXXXXXXXXX	18
2	PRATIK	BIHAR	XXXXXXXXXX	19
3	RIYANKA	SILIGURI	XXXXXXXXXX	20
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19

Table: Student_Courses

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

Join Operators Example

Output:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID
1	HARSH	DELHI	xxxxxxxxxx	18	1
2	PRATIK	BIHAR	xxxxxxxxxx	19	2
3	PRIYANKA	SILIGURI	xxxxxxxxxx	20	2
4	DEEP	RAMNAGAR	xxxxxxxxxx	18	3
5	SAPTARHI	KOLKATA	xxxxxxxxxx	19	1

Types of Operators: Join

1.1 Inner Join

Combines rows from two relations based on a matching condition and only returns rows where there is a match in both relations. If a record in one relation doesn't have a corresponding match in the other, it is excluded from the result. The following are the most common type of inner join.

a. **Conditional Join:** Conditional join or Theta join is a type of inner join in which tables are combined based on the specified condition. In conditional join, the join condition can include <, >, <=, >=, ≠ operators in addition to the '=' operator.

$$Employees \bowtie_{Salary<50000} Departments$$

Example: Joining Employees and Departments on *DepartmentID* where Salary > 50000

b. **Equi Join:** An equi join is a type of inner join where the join condition uses the equality operator ('=') between columns.

$$A \bowtie_{A.Column\ B = C.Column\ B} (C)$$

Example: Joining A and C on *ID* where both relations have this column, returning only matching records.

c. **Natural Join:** A natural join automatically combines relations based on columns with the same name and type, removing duplicate columns in the result. It's a more efficient way of joining.

$$Students \bowtie Enrollments$$

Example: Joining Students and Enrollments where *StudentID* is common in both, and the result contains only unique columns.

Types of Operators: Join

1.2. Outer Join

Outer join is a type of join that retrieves matching as well as non-matching records from related tables. There are three types of outer join:

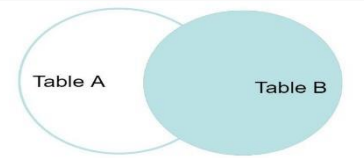
a. **Left Outer Join:** It is also called left join. This type of outer join retrieves all records from the left table and retrieves matching records from the right table.

$A \bowtie B$

Example: Joining Employees with Departments using a left outer join ensures all employees are listed, even those who aren't assigned to any department, with NULL values for the department columns

b. **Right Outer join:** It is also called a right join. This type of outer join retrieves all records from the right table and retrieves matching records from the left table. And for the record which doesn't lies in Left table will be marked as NULL in result Set.

$A \bowtie B$



c. **Full Outer Join:** FULL JOIN creates the result set by combining the results of both LEFT JOIN and RIGHT JOIN. The result set will contain all the rows from both tables. For the rows for which there is no matching, the result set will contain *NULL* values.

$A \bowtie B$

Types of Operators

2. Set Intersection(∩)

Set Intersection basically allows to fetches only those rows of data that are common between two sets of relational tables. Set Intersection in relational algebra is the same set intersection operation in set theory.

Eg: Consider the following table of Students having different optional subjects in their course.

R French

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

R German

Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25

$\pi(Student_Name)(FRENCH) \cap \pi(Student_Name)(GERMAN)$ *Output>>>>*

Student_Name
Vivek
Geeta

Types of Operators

3. Division (÷)

The Division Operator is used to find tuples in one relation that are related to all tuples in another relation. It's typically used for "for all" queries.

Example: Query is to find students who are enrolled in all courses listed in the Course table. In this case, students must be enrolled in both C1 and C2.

Student_Course (Dividend Table):

Student_ID	Course_ID
101	C1
101	C2
102	C1
103	C1
103	C2

Course (Divisor Table):

Course_ID
C1
C2

$Student_Course_{(Student_ID,\ Course_ID)} \div Course_{(Course_ID)}$

Output >>>>>

Student_ID
101
103

Relational Calculus

Relational calculus is a non-procedural query language focuses on what data to retrieve, rather than how to retrieve it, making it different from relational algebra, which is procedural.

- Instead of specifying the exact step-by-step algorithms to fetch data, it focuses purely on what data needs to be retrieved.
- In relational calculus, queries are expressed using logical formulas that describe the desired result, without specifying the exact steps to get there.
- It forms the theoretical foundation for declarative languages like SQL.

There are two types of Relational Calculus

i. Tuple Relational Calculus(TRC)

ii. Domain Relational Calculus(DRC)

Tuple Relational Calculus (TRC)

Tuple Relational Calculus (TRC) is a non-procedural query language used to retrieve data from relational databases by describing the properties of the required data (not how to fetch it).

- It is based on first-order predicate logic and uses tuple variables to represent rows of tables.

Syntax: *{ t | P(t) }*

t: Tuple variable (row placeholder)
P(t): Predicate condition to satisfy
{}: Denotes a set of result tuples

Domain Relational Calculus (DRC)

Domain Relational Calculus (DRC) is a formal query language for relational databases that describes queries by specifying a set of conditions or formulas that the data must satisfy. These conditions are written using domain variables and predicates, and it returns a relation that satisfies the specified conditions.

A general form of a DRC query is written as:

$$\{ < x_1, x_2, x_3, \ldots, x_n > \mid P(x_1, x_2, x_3, \ldots, x_n) \}$$

where, <x1, x2, x3, ..., xn> represents resulting domains variables and P (x1, x2, x3, ..., xn) represents the condition or formula equivalent to the Predicate calculus.

Thank you!!