SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

DBMS Introduction

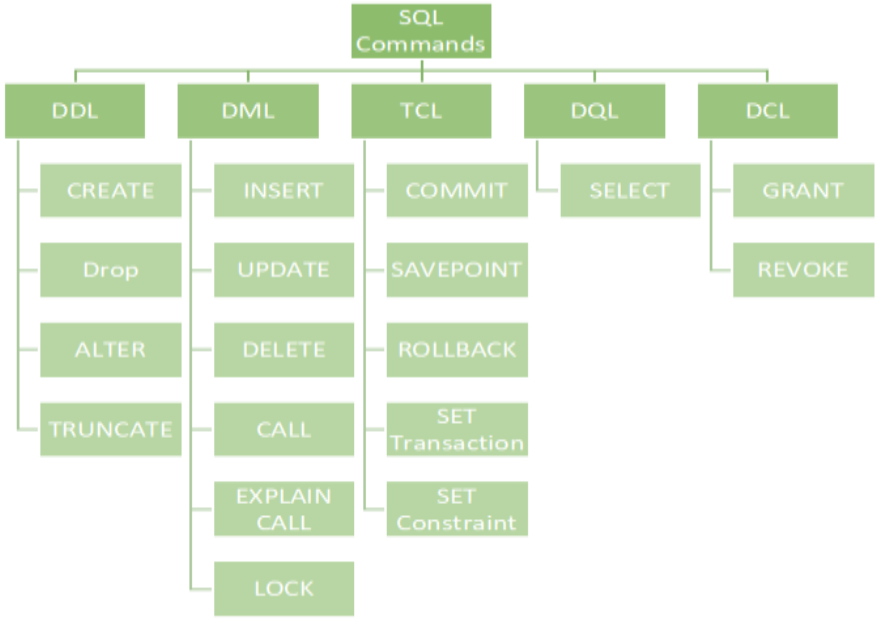
Course Name : 19CAT609 - DATA BASE MANAGEMENT SYSTEM

Class : I Year / II Semester

Unit II – Queries (DDL DML TCL DCL)

Agenda

- **SQL | DDL, DQL, DML, DCL and TCL Commands**
- **Join Types Examples**



SQL | DDL, DQL, DML, DCL and TCL Commands

Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like CREATE, DROP, INSERT, etc. to carry out the required tasks.

SQL commands are like instructions to a table. It is used to interact with the database with some operations. It is also used to perform specific tasks, functions, and queries of data. SQL can perform various tasks like creating a table, adding data to tables, dropping the table, modifying the table, set permission for users.

SQL commands

These SQL commands are mainly categorized into five categories:

DDL – Data Definition Language

DQL – Data Query Language

DML – Data Manipulation Language

DCL – Data Control Language

TCL – Transaction Control Language

DDL (Data Definition Language)

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

DDL (Data Definition Language)

List of DDL commands:

CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).

DROP: This command is used to delete objects from the database.

ALTER: This is used to alter the structure of the database.

TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.

COMMENT: This is used to add comments to the data dictionary.

RENAME: This is used to rename an object existing in the database.

DQL (Data Query Language)

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

List of DQL:

SELECT: It is used to retrieve data from the database.

DML(Data Manipulation Language)

The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

List of DML commands:

INSERT: It is used to insert data into a table.

UPDATE: It is used to update existing data within a table.

DELETE: It is used to delete records from a database table.

LOCK: Table control concurrency.

CALL: Call a PL/SQL or JAVA subprogram.

EXPLAIN PLAN: It describes the access path to data.

DCL (Data Control Language)

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

List of DCL commands:

GRANT: This command gives users access privileges to the database.

Syntax:

GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

REVOKE: This command withdraws the user's access privileges given by using the GRANT command.

Syntax:

REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

TCL (Transaction Control Language)

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure. You can explore more about transactions here. Hence, the following TCL commands are used to control the execution of a transaction:

BEGIN: Opens a Transaction.

COMMIT: Commits a Transaction.

Syntax:

COMMIT;

ROLLBACK: Rollbacks a transaction in case of any error occurs.

Syntax:

ROLLBACK;

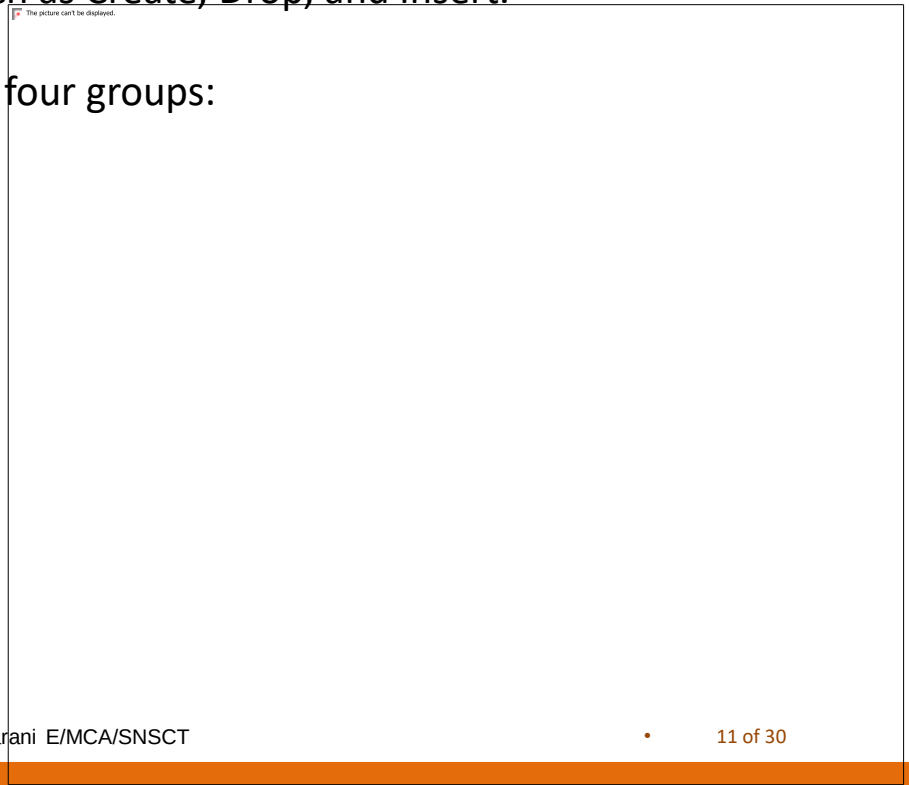
SAVEPOINT: Sets a save point within a transaction.

Syntax:

SAVEPOINT SAVEPOINT_NAME;

SQL Commands

The Structured Query Language (SQL), as we all know, is a database language that allows us to execute specific operations on existing databases as well as construct new databases. To complete the tasks, SQL employs commands such as Create, Drop, and Insert.



These SQL commands are primarily divided into four groups:

Data Definition Language (DDL)
DQL stands for Data Query Language.
DML (Data Manipulation Language)
Data Control Language (DCL)

DDL

1) What Is DDL?

DDL, or data definition language, is a programming language for creating and modifying databases. It's termed a language, but it's more like syntax in and of itself. Alternatively, a sequence of statements allows the user to define or edit data structures and objects like data tables.

How to Use DDL Commands?

As previously stated, DDL is the component of the SQL syntax that deals with the elements of a database by executing commands (also known as statements) such as:

CREATE

ALTER

DROP

RENAME

TRUNCATE

CREATE Statement DDL

1) CREATE Statement

It’s only natural to begin from the beginning, with the CREATE command. It’s used to create complete databases and objects, as the name implies:

CREATE object_type object_name;

For example, if we wish to make a table, we must follow the rules as follows:

CREATE TABLE object_name (column_name data_type);

2) ALTER Statement DDL

The CREATE command demonstrates how the ALTER command works, which is what we use to change existing objects. For example, we can use this DDL statement to change the columns in a table by adding, removing, or renaming them.

To add a column called “date of purchase,” start with the same structure, but instead of using CREATE, use ALTER before naming the object’s type and name. Following that, you write out the specific change:

ALTER TABLE sales

2) ALTER Statement DDL

The CREATE command demonstrates how the ALTER command works, which is what we use to change existing objects. For example, we can use this DDL statement to change the columns in a table by adding, removing, or renaming them.

To add a column called “date of purchase,” start with the same structure, but instead of using CREATE, use ALTER before naming the object’s type and name. Following that, you write out the specific change:

```
ALTER TABLE sales
ADD COLUMN date_of_purchase DATE;
```

3) DROP Statement

What happens if you need to remove a database object? Use the DROP statement in that case:

DROP object_type object_name;

For example, suppose you wish to delete the entire “customers” database using a single line of code:

DROP TABLE customers;

4) RENAME Statement

Another useful DDL tool is RENAME, which allows you to rename an object, such as a database table:

RENAME object_type object name TO new_object_name;

To give you an example, if we hadn't dropped the "customers" table, we could have renamed it "customer data" as follows:

RENAME TABLE customers TO customer_data;

5) TRUNCATE Statement

If we wish to keep using the table as a database object, we can remove data without using DROP. TRUNCATE is the ideal DDL command to use in this situation:

TRUNCATE object_type object_name;

In the context of our example, if we only want to delete the values in our “customers” table, we can use the following code:

TRUNCATE TABLE customers;

2) What is DML?

It’s time to implement DML, or data manipulation language, after you’ve already set up your database – or loaded one that you’d like to work with. This SQL syntax allows you to manipulate existing data objects using a set of actions.

What Are the SQL DML Commands?
DML has various instructions – or statements – that we utilize as we work our way through data tables, as do all SQL syntax components:

SELECT
INSERT
UPDATE
DELETE

The instructions themselves contain a number of keywords or subqueries that indicate the operation we wish to do on the database.

DML

1) SELECT Statement

Starting with the SELECT command, which is used to get data from database objects such as tables:

```
SELECT * FROM sales;
```

2) INSERT Statement

Inserting data into tables is done with the INSERT command. It allows you to add extra records or rows to the table while working with it. The keywords INTO and VALUES go hand in hand with this clause. Consider the following scenario:

```
INSERT INTO sales (purchase_number, date_of_purchase) VALUES
(1, '20xx-xx-xx');
```

DML

3) UPDATE Statement

In SQL, we can select and insert data, but we can also update it. You can use the DML UPDATE command to update existing data in your tables. It has a somewhat distinct syntax that is best explained with an example.

The following code will allow us to change the previously inserted date of purchase number 1 – October 11, 2020 – to something else. That would be December 12, 2022 in our case:

```
UPDATE sales
SET date_of_purchase = '2022-12-12'
WHERE purchase_number = 1;
```

4) DELETE Statement

The DELETE command is identical to DDL's TRUNCATE, but there is one significant distinction. While TRUNCATE allows us to delete all of the records in a table, DELETE allows you to specify exactly what you want to delete.

The following line of code, for example, will delete all records from the "sales" table:

```
DELETE FROM sales;
```

3) What Is DCL?

The data control language (DCL) is essentially a SQL syntax that allows you to manage users' access in a database using a pair of commands. Furthermore, database administrators can manage user access if they have full access to a database.

What Are the DCL Commands?

DCL has only two SQL statements:

GRANT

REVOKE

DCL

1) GRANT Statement

GRANT grants users certain permissions. You can use the following syntax to run the command:

GRANT type_of_permission ON database_name.table_name TO '@username'@'localhost'

You can grant a certain type of permission using this line of code, such as full or partial access to the resources in a specific data table.

2) REVOKE Statement

The REVOKE clause is used to revoke database user permissions and privileges, which is the polar opposite of the GRANT statement. Their syntax, however, is identical:

REVOKE type_of_permission ON database_name.table_name FROM 'username'@'localhost'

Essentially, you can cancel a privilege from someone instead of providing them permission.

TCL

4) What Is TCL?

It is critical to be in control of the transactions you conduct while dealing with relational database management systems in a professional setting. To put it another way, you want to know exactly what you're doing in your database when you insert, delete, or update data.

What Are the TCL Statements?

You must be familiar with TCL's commands in order to deal with it, specifically:

COMMIT

ROLLBACK

SAVEPOINT

It's vital to remember that these will only respond to the DML syntax components INSERT, DELETE, and UPDATE before we look at each one individually.

1) COMMIT Statement

The modifications you've made will be saved permanently in the database, and other users will be able to access the amended version.

Assume you want to edit a record in the "Customers" table by altering the fourth client's last name from Akshay to Amit:

```
UPDATE customers
SET last_name = 'Amit'
WHERE customer_id = 4;
```

Your task isn't over yet, though. The rest of the database system's users won't be able to tell that you've changed anything. You must add a COMMIT statement at the conclusion of the UPDATE block to finish the process:

```
UPDATE customers
SET last_name = 'Amit'
WHERE customer_id = 4
COMMIT;
```

COMMIT basically saves the transaction to the database. The changes are permanent and cannot be reversed once they have been made.

2) ROLLBACK Statement

The number of committed nations might swiftly grow. If you're the administrator, for example, you might need to use COMMIT 20 times today.

As a result, you may inadvertently insert or alter some of your data. Fortunately, the transaction control language function ROLLBACK allows you to restore the database to its previous committed state, undoing any changes you don't wish to retain permanently.

To use this feature, add ROLLBACK; to the end of your code:

```
UPDATE customers
SET last_name = 'Amit'
WHERE customer_id = 4
COMMIT;

ROLLBACK;
```

3) SAVEPOINT Statement

3) SAVEPOINT Statement

Within a transaction, a save point is a logical rollback point. If an error happens after you create a save point, you can use the rollback feature to undo the activities you've done up to that point.

SAVEPOINT savepoint_name;

SQL | Join (Inner, Left, Right and Full Joins)

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL JOIN
NATURAL JOIN
Consider the two tables below as follows:

Student

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

SQL | Join (Inner, Left, Right and Full Joins)

StudentCourse

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

SQL | Join (Inner, Left, Right and Full Joins)

The simplest Join is INNER JOIN.

A. INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table
matching_column: Column common to both the tables.
Note: We can also write JOIN instead of INNER JOIN. JOIN is same as INNER JOIN.

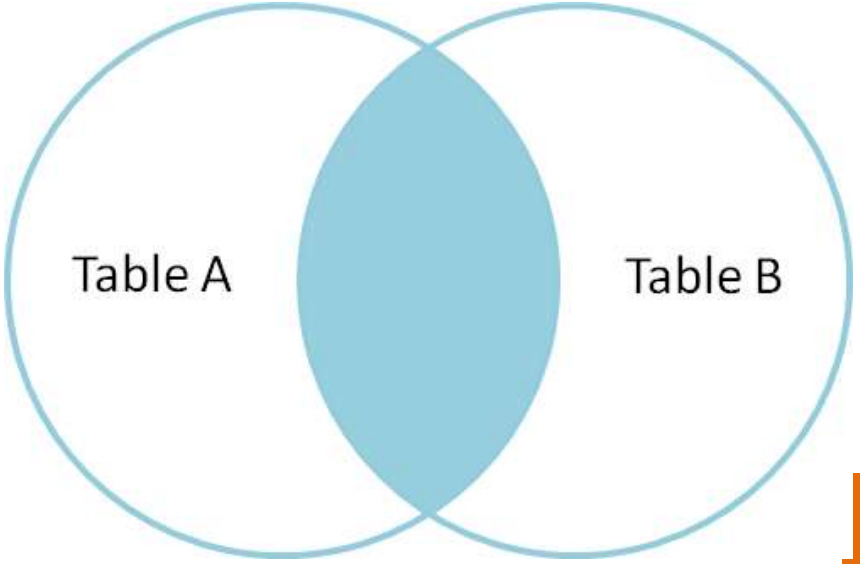
Example Queries(INNER JOIN)

This query will show the names and age of students enrolled in different courses.

```
SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE FROM Student
INNER JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Output:

COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19



B. LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table
matching_column: Column common to both the tables.
Note: We can also use LEFT OUTER JOIN instead of LEFT JOIN, both are the same.

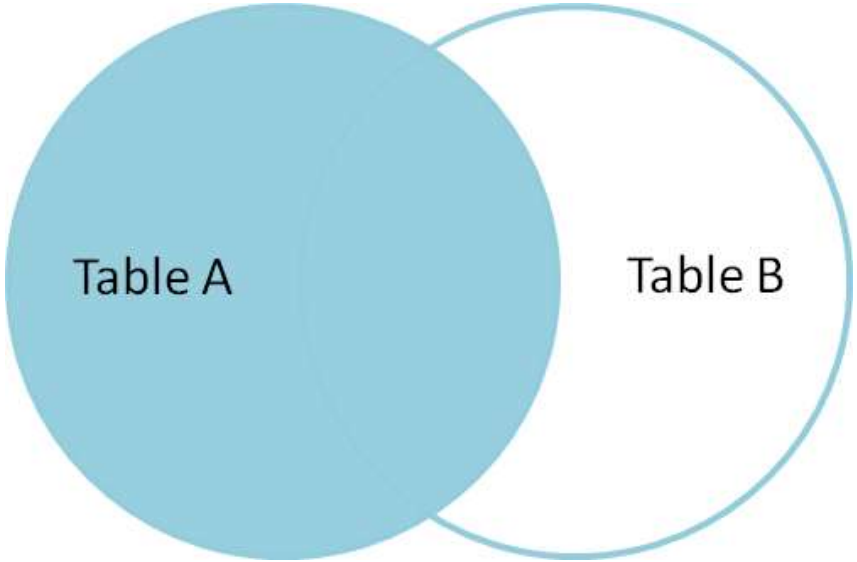
B. LEFT JOIN

Example Queries(LEFT JOIN):

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*



C. RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table
matching_column: Column common to both the tables.
Note: We can also use RIGHT OUTER JOIN instead of RIGHT JOIN, both are the same.

C. RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table
matching_column: Column common to both the tables.
Note: We can also use RIGHT OUTER JOIN instead of RIGHT JOIN, both are the same.

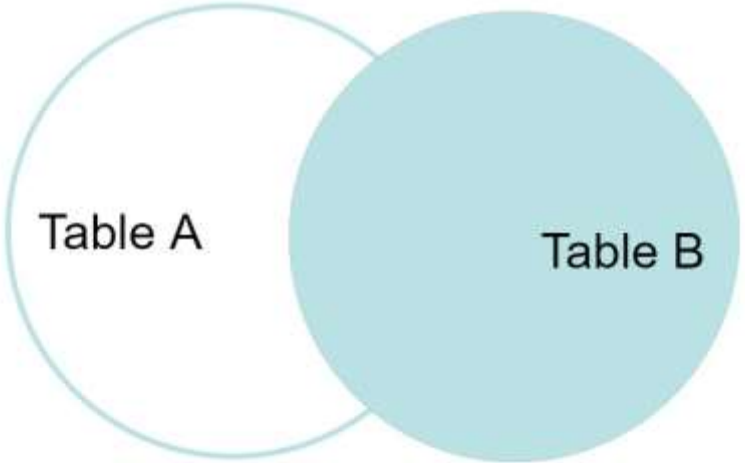
C. RIGHT JOIN

Example Queries(RIGHT JOIN):

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4



D. FULL JOIN

D. FULL JOIN

FULL JOIN creates the result-set by combining 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.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

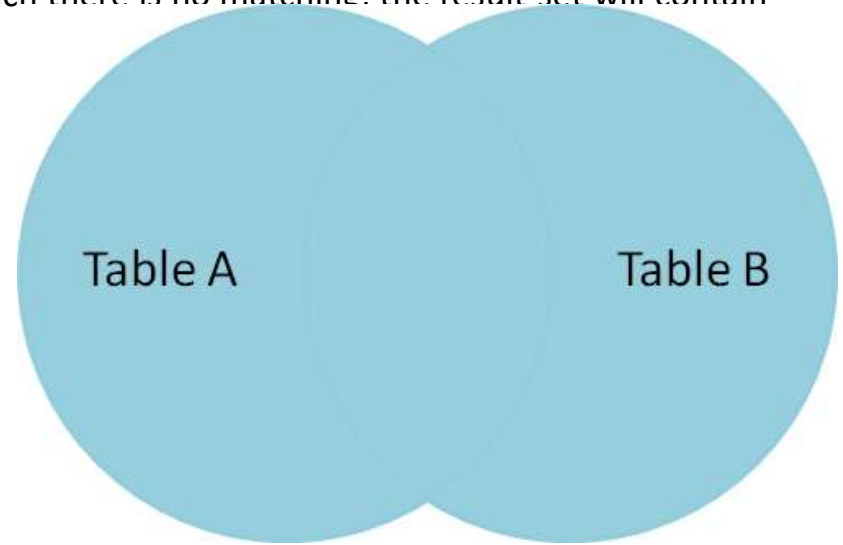


table1: First table.
table2: Second table
matching_column: Column common to both the tables.

Example Queries(FULL JOIN):

Example Queries(FULL JOIN):

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
FULL JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	NULL
ROHIT	NULL
NIRAJ	NULL
NULL	4
NULL	5
NULL	4

E. Natural join (⋈)

E. Natural join (⋈)

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.

Both table must have at list one common column with same column name and same data type.

The two table are joined using Cross join.

DBMS will look for a common column with same name and data type Tuples having exactly same values in common columns are kept in result.

Example:

E. Natural join (⋈)

E. Natural join (⋈)

Employee		
Emp_id	Emp_name	Dept_id
1	Ram	10
2	Jon	30
3	Bob	50

Department	
Dept_id	Dept_name
10	IT
30	HR
40	TIS

E. Natural join (⋈)

Query: Find all Employees and their respective departments.

Solution: (Employee) ⋈ (Department)

Emp_id	Emp_name	Dept_id	Dept_id	Dept_name
1	Ram	10	10	IT
2	Jon	30	30	HR
Employee data			Department data	

References

1. https://k21academy.com/microsoft-azure/data-engineer/sql-commands/
2. https://www.javatpoint.com/dbms-sql-command
3. https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/
4. https://www.geeksforgeeks.org/sql-ddl-dml-tcl-dcl/