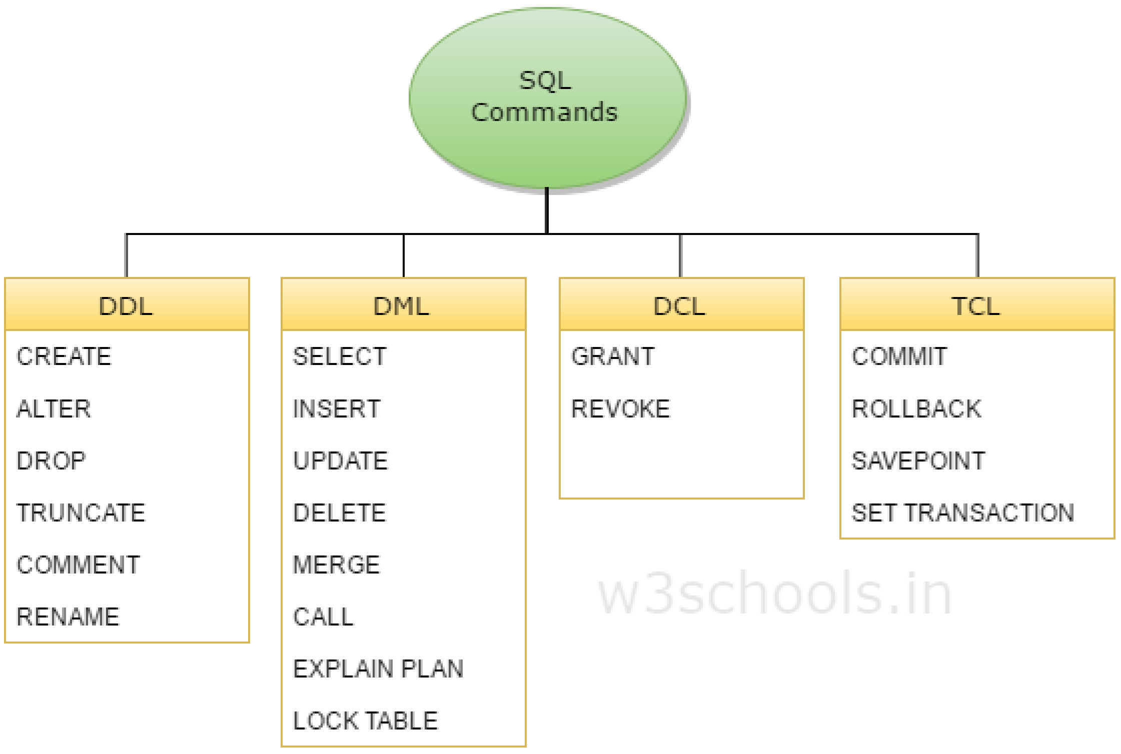
SQL Primer

Agenda

- CRUD with SQL
- JOIN

> **Note**
>
> The script to create the database and insert data can be found here.
> Once your database is set up, download the worksheet from here and start writing some queries! If you get stuck, you can find the answers here.

SQL Commands



DDL

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- **CREATE** – to create database and its objects like (table, index, views, store procedure, function and triggers).
- **ALTER** – alters the structure of the existing database.
- **DROP** – delete objects from the database.
- **TRUNCATE** – remove all records from a table; also, all spaces allocated for the records are removed.
- **COMMENT** – add comments to the data dictionary.
- **RENAME** – rename an object.

DML

DML is short name of Data Manipulation Language which deals with data manipulation, and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE etc, and it is used to store, modify, retrieve, delete and update data in database.

- **SELECT** – retrieve data from one or more tables.
- **INSERT** – insert data into a table.
- **UPDATE** – updates existing data within a table.
- **DELETE** – delete all records from a table.
- **MERGE** – UPSERT operation (insert or update)
- **CALL** – call a PL/SQL or Java subprogram.
- **EXPLAIN PLAN** – interpretation of the data access path.
- **LOCK TABLE** – concurrency control.

DCL

DCL is short name of Data Control Language which includes commands such as GRANT, and mostly concerned with rights, permissions and other controls of the database system.

- **GRANT** – allow users access privileges to database.
- **REVOKE** – withdraw users access privileges given by using the GRANT command.

TCL

TCL is short name of Transaction Control Language which deals with transaction within a database.

- **COMMIT** – commits a transaction.
- **ROLLBACK** – rollback a transaction in case of any error occurs.
- **SAVEPOINT** – a point inside a transaction that allows rollback state to what it was at the time of the savepoint.
- **SET TRANSACTION** – specify characteristics for the transaction.

CRUD with SQL

Create rows

Keyword: `INSERT` **Syntax**: `INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...)`

Examples

1. Insert a row with all columns

```
    INSERT INTO students VALUES (1, 'Tantia', 'Tope', 't@t.com',
'1234567890', 1);
```

2. Insert a row with some columns

```
    INSERT INTO students (first_name, last_name) VALUES ('Tantia',
'Tope');
```

Read rows

Keyword: `SELECT` **Syntax**: `SELECT column1, column2, ... FROM table_name WHERE condition ORDER BY column1, column2, ... ASC/DESC LIMIT #`

Examples

1. Get all rows

   ```
   SELECT * FROM students;
   ```

2. Get certain fields from all rows

   ```
   SELECT first_name, last_name FROM students;
   ```

3. Filter rows by condition

   ```
   SELECT * FROM students WHERE first_name = 'Tantia';
   ```

4. Order rows by column

   ```
   SELECT * FROM students ORDER BY first_name ASC;
   ```

5. Limit number of rows

   ```
   SELECT * FROM students LIMIT 10;
   ```

Common operators for WHERE clause

Operator	Description	Example
=	Equal	`SELECT * FROM students WHERE first_name = 'Tantia'`
!= or <>	Not equal	`SELECT * FROM students WHERE first_name != 'Tantia'`
NOT	NOT	`SELECT * FROM students WHERE NOT first_name = 'John'`

Operator	Description	Example
>	Greater than	`SELECT * FROM students WHERE iq > 150`
<	Less than	`SELECT * FROM students WHERE age < 100`
>=	Greater than or equal	`SELECT * FROM students WHERE age >= 18`
<=	Less than or equal	`SELECT * FROM students WHERE age <= 18`
AND	AND	`SELECT * FROM students WHERE first_name = 'Tantia' AND last_name = 'Tope'`
OR	OR	`SELECT * FROM students WHERE first_name = 'John' OR last_name = 'Mycroft'`
IN	IN	`SELECT * FROM students WHERE first_name IN ('John', 'Mycroft')`
BETWEEN	BETWEEN	`SELECT * FROM students WHERE iq BETWEEN 100 AND 150`
LIKE	LIKE	`SELECT * FROM students WHERE first_name LIKE '%T%'`
REGEXP	REGEXP	`SELECT * FROM students WHERE first_name REGEXP '^[A-Z]{1}'`
NULL	NULL	`SELECT * FROM students WHERE first_name IS NULL`
NOT NULL	NOT NULL	`SELECT * FROM students WHERE first_name IS NOT NULL`

String matching wildcards

With `LIKE` you can use the following two wildcard characters in the pattern:

- `%` matches any number of characters, even zero characters.
- `_` matches exactly one character.

```sql
SELECT * FROM students WHERE first_name LIKE 'T%';
```

```sql
SELECT * FROM students WHERE first_name LIKE 'T_';
```

Update rows

Keyword: `UPDATE` **Syntax**: `UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition`

Examples

1. Update a row

```sql
UPDATE students SET first_name = 'Tantia' WHERE id = 1;
```

2. Update a row with a condition

```sql
UPDATE students SET first_name = 'Tantia' WHERE id = 1 AND first_name
= 'John';
```

3. Update multiple columns

```sql
UPDATE students SET first_name = 'Tantia', last_name = 'Tope' WHERE id
= 1 AND first_name = 'John';
```

Delete rows

Keyword: `DELETE` **Syntax**: `DELETE FROM table_name WHERE condition`

Examples

1. Delete a row with a condition

```sql
DELETE FROM students WHERE id = 1 AND first_name = 'John';
```

2. Delete a multiple rows

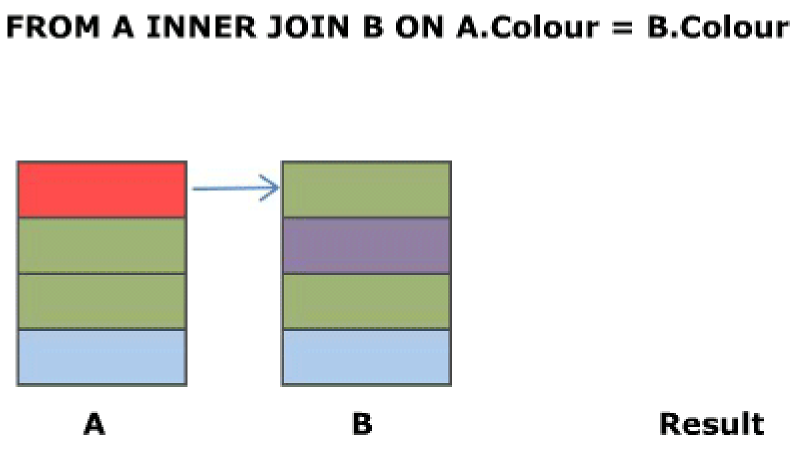
```sql
DELETE FROM students WHERE id IN (1, 2, 3);
```

Join

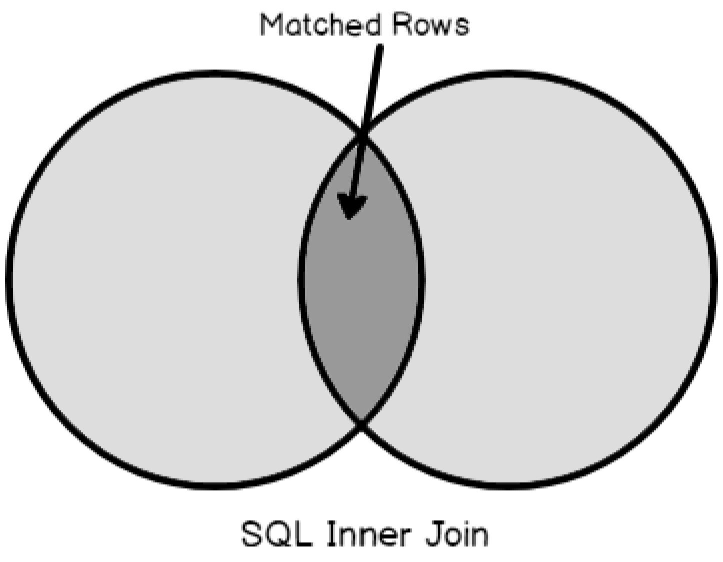
Join is the widely-used clause in the SQL Server essentially to combine and retrieve data from two or more tables. In a real-world relational database, data is structured in many tables and which is why, there is a constant need to join these multiple tables based on logical relationships between them.

Inner join

Inner Join clause in SQL Server creates a new table (not physical) by combining rows that have matching values in two or more tables. This join is based on a logical relationship (or a common field) between the tables and is used to retrieve data that appears in both tables.



Assume, we have two tables, Table A and Table B, that we would like to join using SQL Inner Join. The result of this join will be a new result set that returns matching rows in both these tables. The intersection part in black below shows the data retrieved using Inner Join.



Keyword: `INNER JOIN` or simply `JOIN` **Syntax**: `SELECT column1, column2, ... FROM table_name1 JOIN table_name2 ON condition`

For example, we want to get the batch names of all the students along with their names.

id	first_name	last_name	batch_name
1	John	Watson	Sherlock
2	Mycroft	Holmes	Sherlock

This can be achieved by using the following SQL query:

```
SELECT s.first_name, s.last_name, b.batch_name FROM students s JOIN
batches ON s.batch_id = b.id;
```