Parul® University

Unit 2- SQL

Subject Code: 303105203

Prof. S.W.Thakare
Assistant Professor,
Computer science & Engineering

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-2

SQL

Structured Query Language(SQL)

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

SQL Commands

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

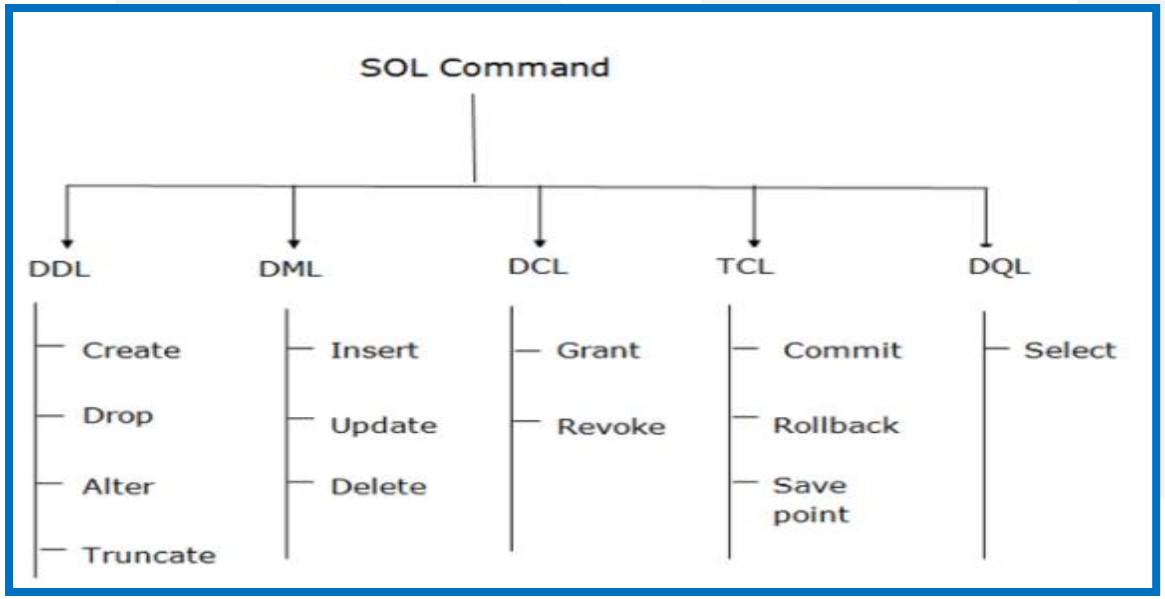

Figure: 1.24 SQL Commands

(**Image Source** : https://www.google.com/imgres?imgurl=https%3A%2F%2Fstatic.javatpoint.com%2Fdbms%2Fimages%2Fdbms-sql-command.png&imgrefurl=http)

Data Definition Language (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:

1. CREATE
2. ALTER
3. DROP
4. TRUNCATE

Data Definition Language (DDL Command)

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

- **Syntax:**
  - CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

- **Example:**
  - CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

Data Definition Language (DDL Command)

2. **DROP:** It is used to delete both the structure and record stored in the table.

- **Syntax**
  - DROP TABLE ;
- **Example**
  - DROP TABLE EMPLOYEE;

Data Definition Language (DDL Command)

3. **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:**
- **To add a new column in the table**
- ALTER TABLE table_name ADD column_name COLUMN-definition;
- **To modify existing column in the table:**
- ALTER TABLE MODIFY(COLUMN DEFINITION....);
- **EXAMPLE:**
- ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
- ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

Data Definition Language (DDL Command)

4. **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;

Data Manipulation Language(DML)

- 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:

1. INSERT
2. UPDATE
3. DELETE

Data Manipulation Language(DML Commands)

1. **INSERT:** The INSERT statement is a SQL query. It is 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);

- **For example:**

– INSERT INTO BOOK (Author, Subject) VALUES ("Shital", "DBMS");

Data Manipulation Language(DML Commands)

2. **UPDATE:** This command is used to update or modify the value of a column in the table.

- **Syntax:**

– UPDATE table_name SET [column_name1= value1,...column_nameN = value N] [WHERE CONDITION]

- **For example:**

– UPDATE students SET User_Name = 'Shital WHERE Student_Id = '3'

Data Manipulation Language(DML Commands)

3. **DELETE:** It is used to remove one or more row from a table.

- **Syntax:**
  - DELETE FROM table_name [WHERE condition];
- **For example:**
  - DELETE FROM Book WHERE Author="Shital";

Data Control Language(DCL)

- DCL commands are used to grant and take back authority from any database user.
- Here are some commands that come under DCL:

1. Grant
2. Revoke

Data Control Language(DCL Commands)

1. **Grant:** It is used to give user access privileges to a database.

- **Example**

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

2. **Revoke:** It is used to take back permissions from the user.

- **Example**

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

Transaction Control Language(TCL)

- 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:

1. COMMIT
2. ROLLBACK
3. SAVEPOINT

Transaction Control Language(TCL Commands)

1. **Commit:** Commit command is used to save all the transactions to the database.

- **Syntax:-**COMMIT;
- **Example:**
  - DELETE FROM CUSTOMERS WHERE AGE = 25;
  - COMMIT;

Transaction Control Language(TCL Commands)

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

- **Syntax:-**ROLLBACK;
- **Example:**
  - DELETE FROM CUSTOMERS WHERE AGE = 25;
  - ROLLBACK;

Transaction Control Language(TCL Commands)

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

- **Syntax:-**SAVEPOINT SAVEPOINT_NAME;

Domain Types in SQL

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n*.
- **varchar(n).** Variable length character strings, with user-specified maximum length *n*.
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point. (ex., **numeric**(3,1), allows 44.5 to be stores exactly, but not 444.5 or 0.32)
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.

Domain Types in SQL

- An SQL relation is defined using the **create table** command:

  create table r (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

  - r is the name of the relation
  - each A_i is an attribute name in the schema of relation r
  - D_i is the data type of values in the domain of attribute A_i

- Example:

```
create table instructor (
    ID          char(5),
    name        varchar(20),
    dept_name   varchar(20),
    salary      numeric(8,2))
```

Update Table in SQL

Updates to tables

- **Insert**
  - **insert into** *instructor* **values** ('10211', 'Smith', 'Biology', 66000);
- **Delete**
  - Remove all tuples from the *student* relation
    - **delete from** *student*
- **Drop Table**
  - **drop table** *r*
- **Alter**
  - **alter table** *r* **add** *A D*
    - where *A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A.*
    - All exiting tuples in the relation are assigned *null* as the value for the new attribute.
  - **alter table** *r* **drop** *A*
    - where *A* is the name of an attribute of relation *r*
    - Dropping of attributes not supported by many databases.

SQL

Basic Query Structure

- A typical SQL query has the form:

select $A_1, A_2, \ldots, A_n$
from $r_1, r_2, \ldots, r_m$
where P

  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate.
- The result of an SQL query is a relation.

SQL

The select Clause

- The **select** clause lists the attributes desired in the result of a query
  - corresponds to the projection operation of the relational algebra
- Example: find the names of all instructors:

  select *name*
  from *instructor*
- NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)
  - E.g., *Name* ≡ *NAME* ≡ *name*
  - Some people use upper case wherever we use bold font.

SQL

- SQL allows duplicates in relations as well as in query results.
- To force the elimination of duplicates, insert the keyword **distinct** after select.
- Find the department names of all instructors, and remove duplicates

```
select distinct dept_name
from instructor
```

- The keyword **all** specifies that duplicates should not be removed.

```
select all dept_name
from instructor
```

SQL

- An asterisk in the select clause denotes "all attributes"

```
select *
from instructor
```

- An attribute can be a literal with no **from** clause

```
select '437'
```

  - Results is a table with one column and a single row with value "437"
  - Can give the column a name using:

```
select '437' as FOO
```

- An attribute can be a literal with **from** clause

```
select 'A'
from instructor
```

  - Result is a table with one column and *N* rows (number of tuples in the *instructors* table), each row with value "A"

SQL

- The **select** clause can contain arithmetic expressions involving the operation, +, –, *, and /, and operating on constants or attributes of tuples.
  - The query:

    select *ID, name, salary/12*
    from *instructor*

    would return a relation that is the same as the *instructor* relation, except that the value of the attribute *salary* is divided by 12.
  - Can rename "*salary/12*" using the **as** clause:

    select *ID, name, salary/12* **as** *monthly_salary*

SQL

The where Clause

- The **where** clause specifies conditions that the result must satisfy
  - Corresponds to the selection predicate of the relational algebra.
- To find all instructors in Comp. Sci. dept

```
select name
from instructor
where dept_name = 'Comp. Sci.'
```

- Comparison results can be combined using the logical connectives **and, or,** and **not**
  - To find all instructors in Comp. Sci. dept with salary > 80000

SQL

The from Clause

- The **from** clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra.
- Find the Cartesian product *instructor X teaches*

```
select *
from instructor, teaches
```

  - generates every possible instructor – teaches pair, with all attributes from both relations.
  - For common attributes (e.g., *ID*), the attributes in the resulting table are renamed using the relation name (e.g., *instructor.ID*)
- Cartesian product not very useful directly, but useful combined with where-clause condition (selection operation in relational algebra).

SQL

Modification of the Database

- Deletion of tuples from a given relation.
- Insertion of new tuples into a given relation
- Updating of values in some tuples in a given relation

SQL

Insertion

- Add a new tuple to *course*

```
insert into course
    values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

- or equivalently

```
insert into course (course_id, title, dept_name, credits)
    values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

- Add a new tuple to *student* with *tot_creds* set to null

```
insert into student
    values ('3003', 'Green', 'Finance', null);
```

SQL

Deletion

- Delete all instructors

```
delete from instructor
```

- Delete all instructors from the Finance department

```
delete from instructor
where dept_name= 'Finance';
```

- Delete all tuples in the *instructor* relation for those instructors associated with a department located in the Watson building.

```
delete from instructor
where dept name in (select dept name
                    from department
                    where building = 'Watson');
```

SQL

Updates

- Increase salaries of instructors whose salary is over $100,000 by 3%, and all others by a 5%
  - Write two **update** statements:

```
update instructor
    set salary = salary * 1.03
    where salary > 100000;
update instructor
    set salary = salary * 1.05
    where salary <= 100000;
```

  - The order is important
  - Can be done better using the **case** statement (next slide)

References

[1] Abraham Silberschatz, Henry F. Korth and S. Sudarshan, Database System Concepts, McGraw-Hill Education (Asia), Seventh Edition, 2019.

[2] C. J. Date, A. Kannan and S. Swamynathan, An Introduction to Database Systems, Pearson Education, Eighth Edition, 2009.

[3] Database Management Systems, CSE, DIET, https://www.darshan.ac.in/DIET/CE/GTU-Computer-Engineering-Study-Material

[4] Database management systems by Raghu Ramakrishnan and Johannes Gehrke http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html

[5] Database management system tutorial, https://www.tutorialspoint.com/dbms/index.htm

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in