Database Management System

Unit-5

Mrs. Kiran Bala Dubey

Assistant Professor

Department of Computer Science

Govt. N. P. G. College of Science,

Raipur

UNIT - V: Query Processing and Security

Introduction to SQL constructs (SELECT…FROM, WHERE… GROUP BY… HAVING… ORDERBY….), INSERT, DELETE, UPDATE, DROP, VIEW definition and use, Temporary tables, Nested queries, and correlated nested queries, Integrity constraints: Not null, unique, check, primary key, foreign key, references, Inner and Outer Joins. **Query Processing:** Parsing, translation, optimization, evaluation and overview of Query Processing. **Protecting the Data Base:** Integrity, Security and Recovery. Domain Constraints, Referential Integrity, Assertion, Triggers, Security & Authorization in SQL.

- SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).
- SQL was the first commercial language introduced for E.F Codd's Relational model of database.
- SQL is used to perform all types of data operations in RDBMS.
- It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
- All the RDBMS like MySQL, Informix, Oracle, MS Access, Sybase and SQL Server use SQL as their standard database language.
- SQL allows users to query the database in a number of ways, using English-like statements.

SQL follows the following rules

- Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.
- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text line.
- Using the SQL statements, you can perform most of the actions in a database.
- SQL depends on tuple relational calculus and relational algebra

Types of Structured Query Language -

- **DDL (Data Definition Language)**

  DDL is used to define table schemas.

- **DML (Data Manipulation Language)**

  DML is used for inserting, updating and deleting data from the database.

- **DCL (Data Control Language)**

  DCL is used for user & permission management. It controls the access to the database.

- **DQL (Data Query Language)**

  Data query language is used to fetch data from tables based on conditions that we can easily apply.

- **TCL (Transaction Control Language)**

  These commands are to keep a check on other commands and their affect on the database. These commands can cancel changes made by other commands by rolling the data back to its original state. It can also make any temporary change permanent.

DDL (Data Definition Language)

Command	Description
Create table	to create new table or database
Alter table	for alteration
Truncate table	delete all data from table
Drop table	to drop a table
Rename table	to rename a table

DML (Data Manipulation Language)

Command	Description
insert	**to insert a new row**
update	**to update existing row**
delete	**to delete a row**

DCL (Data Control Language)

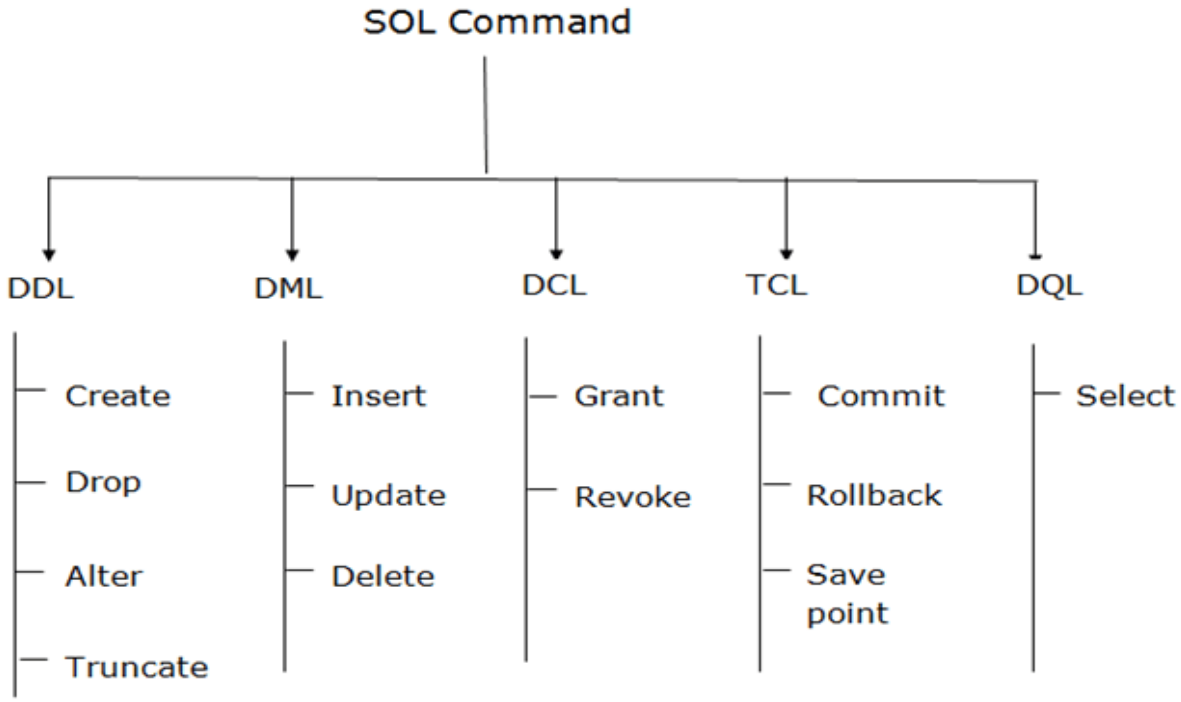
Command	Description
grant	grant permission of right
revoke	take back permission.

DQL (Data Query Language)

Command	Description
select	retrieve records from one or more table

TCL (Transaction Control Language)

Command	Description
commit	**to permanently save**
rollback	**to undo change**
savepoint	**to save temporarily**


SOL Command
DDL
DML
DCL
TCL
DQL
Create
Drop
Alter
Truncate
Insert
Update
Delete
Grant
Revoke
Commit
Rollback
Save point
Select

Integrity constraints -

- Integrity Constraints are used to apply business rules for the database tables.
- Constraints can be defined in two ways
  1) The constraints can be specified immediately after the column definition. This is called **column-level definition**.
  2) The constraints can be specified after all the columns are defined. This is called **table-level definition**.
- NOT NULL Constraint − Ensures that a column cannot have NULL value.
- DEFAULT Constraint − Provides a default value for a column when none is specified.
- UNIQUE Constraint − Ensures that all values in a column are different.
- PRIMARY Key − Uniquely identifies each row/record in a database table.
- FOREIGN Key − Uniquely identifies a row/record in any of the given database table.
- CHECK Constraint − The CHECK constraint ensures that all the values in a column satisfies certain conditions.

DDL

- **CREATE TABLE TableName (Column1 datatype,**
  Column2 datatype,
  Column3 datatype,
  ....
  ColumnN datatype);

```
CREATE TABLE student ( Roll_no number (6),
                       Name varchar2(20),
                       Class varchar2(10),
                       C number(3),
                       DBMS number(3),
                       SO number(3));
```

- **DROP TABLE TableName;**

  DROP TABLE student;

- **ALTER TABLE TableName ADD (ColumnName Datatype);**
- **ALTER TABLE TableName DROP COLUMN (ColumnName);**
- **ALTER TABLE MODIFY(ColumnName NewDataType(size));**

  ALTER TABLE student ADD (DOB date);

  ALTER TABLE student DROP COLUMN (DOB);

  ALTER TABLE MODIFY(Name varchar2(15));

- **DESCRIBE TableName;**

  DESC student;

DML

- **INSERT INTO TABLE_NAME**

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

  INSERT INTO student values(101,'Sumit', 'BCA2',56,65,67);

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

  UPDATE student SET Name = 'Amit' WHERE Roll_no=101;

- **DELETE FROM table_name [WHERE condition];**

  DELETE From student where Roll_no=101;

- **SELECT expressions FROM TABLES WHERE conditions;**

```
SELECT * FROM student;

SELECT * FROM student WHERE DBMS>60;
```

- **The 'ORDER BY' Statement –** It is used to sort the required results in ascending or descending order. The results are sorted in ascending order by default.

SELECT Column1, Column2, ...ColumnN FROM TableName ORDER BY Column1, Column2, ... ASC|DESC;

```
SELECT name FROM student ORDER BY name;
```

- **The 'GROUP BY' Statement -** It is used with the aggregate functions to group the result-set by one or more columns.

SELECT Column1, Column2,..., ColumnN FROM TableName WHERE Condition GROUP BY ColumnName(s);

```
SELECT name, class FROM student GROUP BY class;
```

- **The 'HAVING' Clause -** It is used in SQL because the **WHERE keyword** cannot be used everywhere.

```
SELECT class, count(*) FROM student GROUP BY class HAVING class='BCA2';
```

- **Aggregate Functions**:
  - MIN()
  - MAX()
  - COUNT()
  - SUM()
  - AVG()
- **Nested queries** are those queries which have an outer query and inner subquery. So, basically, the subquery is a query which is nested within another query.

```
SELECT * FROM student WHERE DBMS = (SELECT MAX(DBMS) FROM student);
```

- A **view** in SQL is a imaginary table, which is derived from other tables. So, a view contains rows and columns similar to a real table and has fields from one or more table.

CREATE VIEW ViewName AS SELECT Column1, Column2, ..., ColumnN FROM TableName WHERE Condition;

CREATE VIEW PGDCA AS SELECT Roll_no, name FROM student WHERE class = ‘PGDCA’;

Create a table **Student** with following fields-

Roll_no	Number	6
Name	varchar2	20
Class	varchar2	10
C	number	3
DBMS	number	3
SO	number	3

```
CREATE TABLE student ( Roll_no number (6), Name varchar2(20),
                       Class varchar2(10),  C number(3),
                       DBMS number(3),  SO number(3));
```

Q.1- List all records from the table student.

SELECT * FROM student;

Q.2- List all Name and Class from the table student.

SELECT name, class FROM student;

Q.3- List name of all students of BCA2.

SELECT name FROM student WHERE class='BCA2';

Q.4- List the name of students class-wise.

SELECT name, class FROM student GROUP BY class;

Q.5- List the name of students in ascending order.

SELECT name FROM student ORDER BY name ASC;

Q.6- Display Roll_no, Name, and total marks obtained by each student.

SELECT roll_no, name, c+dbms+so FROM student;

Q.7- Display the records of those students who obtained more than 40 marks in DBMS.

SELECT * FROM student WHERE dbms>40;

Q.8- Display the records of BCA2 students who obtained more than 40 marks in C.

SELECT * FROM student WHERE class='BCA2' AND c>40;

Q.9- Display the Name, Class and marks of C with grace 5 of student who obtained less than 40 marks in C.

SELECT name, class, c+5 FROM student WHERE c <40;

Q. 10- Display the Name of all students which starts from letter 's';

SELECT name FROM student WHERE name LIKE 's%';

SELECT name FROM student WHERE name LIKE '_____';

Q. 11- Display the records of those students who obtained 60 to 80 marks in DBMS.

SELECT * FROM student WHERE dbms BETWEEN 60 AND 80;

Q.12- List all students who obtained 32, 35, 44 or 45 marks in DBMS.

SELECT * FROM student WHERE dbms IN (32,35,44,45);

Q.13- Display the total number of students in each class.

SELECT class, count(*) FROM student GROUP BY class;

Q.14- Display records of those students who obtained highest marks in DBMS.

SELECT * FROM student WHERE dbms=(SELECT MAX(dbms) FROM student);

Q. 15. Display the total number of students who obtained more than 75 marks in DBMS.

SELECT COUNT(*) FROM student WHERE dbms >75;

Q. 16. Display marks in C, class wise where marks in C is greater than 80.

SELECT class, c FROM student GROUP BY class HAVING c>80;

Q. 17. Display name of those students class wise who obtained average marks in C.

SELECT class, name, c FROM student GROUP BY class HAVING c(SELECT AVG(c) FROM student);

- JOINS are used to combine rows from two or more tables, based on a related column between those tables. The following are the types of joins:
- **INNER JOIN:** This join returns those records which have matching values in both the tables.
- **FULL JOIN:** This join returns all those records which either have a match in the left or the right table.
- **LEFT JOIN:** This join returns records from the left table, and also those records which satisfy the condition from the right table.
- **RIGHT JOIN:** This join returns records from the right table, and also those records which satisfy the condition from the left table.

Query Processing

- Query Processing is the activity performed in extracting data from the database. In query processing, it takes various steps for fetching the data from the database. The steps involved are:
- Parsing and translation
- Optimization
- Evaluation

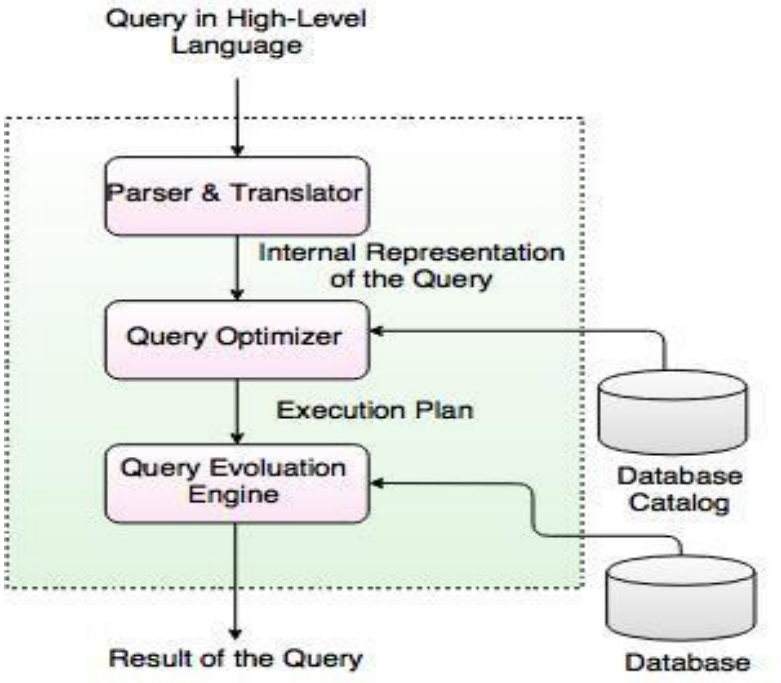


Fig. Query Processing

•The first step is to transform the query into a standard form.
•A query is **translated** into SQL and into a relational algebraic expression. During this process, **Parser** checks the syntax and verifies the relations and the attributes which are used in the query.
•The second step is Query **Optimize**r. In this, it transforms the query into equivalent expressions that are more efficient to execute.
•The third step is Query **evaluation**. It executes the above query execution plan and returns the result.

SELECT Ename FROM Employee WHERE Salary > 5000;

σ Salary > 5000 (π Ename (Employee))

OR

π Ename (σ Salary > 5000 (Employee))

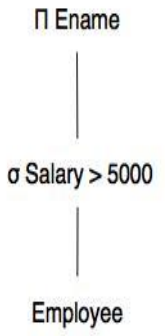


Fig. Query Execution Plan

Protecting the Data Base

- **Database integrity** ensures that data in the database is correct and consistent.
- **Integrity constraints** guard against accidental damage to the database, by ensuring that authorized changes to the database do not result in a loss of data consistency.
- **Referential Integrity** - Ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.
- **Domain constraints** are the most elementary form of integrity constraint. They test values inserted in the database, and test queries to ensure that the comparisons make sense.
- **Recovery -**Database recovery is the process of restoring the database to a correct (consistent) state in the event of a failure. In other words, it is the process of restoring the database to the most recent consistent state that existed shortly before the time of system failure.

- An **assertion** is a predicate expressing a condition that we wish the database always to satisfy.
- An assertion in SQL takes the form

  create assertion <assertion name> check <predicate>
- When an assertion is made, the system tests it for validity, and tests it again on every update that may violate the assertion.
- **Security** - protection from malicious attempts to steal or modify data.
  - Confidentiality, integrity, and availability are the hallmarks of database security.
  - Database system level - Authentication and authorization mechanisms to allow specific users access only to required data .
  - Operating system level - Operating system super-users can do anything they want to the database.
  - Network level - must use encryption to prevent Eavesdropping (unauthorized reading of messages) Masquerading (pretending to be an authorized user or sending messages supposedly from authorized users)

- **Triggers** are a set of SQL statements which are stored in the database catalog. These statements are executed whenever an event associated with a table occurs. So, a trigger can be invoked either BEFORE or AFTER the data is changed by INSERT, UPDATE or DELETE statement.

  CREATE TRIGGER [TriggerName] [BEFORE | AFTER] {INSERT | UPDATE | DELETE} on [TableName] [FOR EACH ROW] [TriggerBody]

Qu. Write a database trigger before insert/update/delete for each statement not allowing any of these operations on the table student on Saturday and Sunday.

Ans.

```
Create or replace trigger t1 before insert or update or delete on student
Begin
If to_char(sysdate,'dy') in ('sat','sun') then
        Raise_application_error(-20410,'Today is holiday so no work……Enjoy');
End if;
If to_char(sysdate,'hh24')<9 or to_char(sysdate,'hh24')>17 then
        Raise_application_error(-20420,'Not a working hour……');
End if;
End;
/
```