23CAT603 DATABASE MANAGEMENT SYSTEMS

UNIT-II RELATIONAL MODEL AND QUERY EVALUATION

Relational Model Concepts – Relational Algebra – SQL – Basic Queries – Complex SQL Queries – Views – Constraints.

16 Marks Questions and Answers

1. Relational Model Concepts

E.F. Codd proposed the relational Model to model data in the form of relations or tables. After designing the conceptual model of the Database using ER diagram, we need to convert the conceptual model into a relational model which can be implemented using any RDBMS language like Oracle SQL, MySQL, etc. So we will see what the Relational Model is.

The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

What is the Relational Model?
The relational model represents how data is stored in Relational Databases. A relational database consists of a collection of tables, each of which is assigned a unique name. Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE, and AGE shown in the table.

Table Student

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

Important Terminologies

Attribute: Attributes are the properties that define an entity. e.g.; ROLL_NO, NAME, ADDRESS

Relation Schema: A relation schema defines the structure of the relation and represents the name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.

Tuple: Each row in the relation is known as a tuple. The above relation contains 4 tuples, one of which is shown as:

1	RAM	DELHI	9455123451	18

Relation Instance: The set of tuples of a relation at a particular instance of time is called a relation instance. Table 1 shows the relation instance of STUDENT at a particular time. It can change whenever there is an insertion, deletion, or update in the database.

Degree: The number of attributes in the relation is known as the degree of the relation. The STUDENT relation defined above has degree 5.

Cardinality: The number of tuples in a relation is known as cardinality. The STUDENT relation defined above has cardinality 4.

Column: The column represents the set of values for a particular attribute. The column ROLL_NO is extracted from the relation STUDENT.

ROLL_NO
1
2
3
4

NULL Values: The value which is not known or unavailable is called a NULL value. It is represented by blank space. e.g.; PHONE of STUDENT having ROLL_NO 4 is NULL.

Relation Key: These are basically the keys that are used to identify the rows uniquely or also help in identifying tables. These are of the following types.

- Primary Key
- Candidate Key
- Super Key
- Foreign Key
- Alternate Key
- Composite Key

Constraints in Relational Model

While designing the Relational Model, we define some conditions which must hold for data present in the database are called Constraints. These constraints are checked before performing any operation (insertion, deletion, and updation) in the database. If there is a violation of any of the constraints, the operation will fail.

Domain Constraints

These are attribute-level constraints. An attribute can only take values that lie inside the domain range. e.g.; If a constraint AGE>0 is applied to STUDENT relation, inserting a negative value of AGE will result in failure.

Key Integrity

Every relation in the database should have at least one set of attributes that defines a tuple uniquely. Those set of attributes is called keys. e.g.; ROLL_NO in STUDENT is key. No two students can have the same roll number. So a key has two properties:

It should be unique for all tuples.

It can't have NULL values.

Referential Integrity

When one attribute of a relation can only take values from another attribute of the same relation or any other relation, it is called referential integrity. Let us suppose we have 2 relations

Table Student

ROLL_NO	NAME	ADDRESS	PHONE	AGE	BRANCH_CODE
1	RAM	DELHI	9455123451	18	CS
2	RAMESH	GURGAON	9652431543	18	CS
3	SUJIT	ROHTAK	9156253131	20	ECE
4	SURESH	DELHI		18	IT

Table Branch

BRANCH_CODE	BRANCH_NAME
CS	COMPUTER SCIENCE
IT	INFORMATION TECHNOLOGY
ECE	ELECTRONICS AND COMMUNICATION ENGINEERING
CV	CIVIL ENGINEERING

BRANCH_CODE of STUDENT can only take the values which are present in BRANCH_CODE of BRANCH which is called referential integrity constraint. The relation which is referencing another relation is called REFERENCING RELATION (STUDENT in this case) and the relation to which other relations refer is called REFERENCED RELATION (BRANCH in this case).

Anomalies in the Relational Model

An anomaly is an irregularity or something which deviates from the expected or normal state. When designing databases, we identify three types of anomalies: Insert, Update, and Delete.

Insertion Anomaly in Referencing Relation

We can't insert a row in REFERENCING RELATION if referencing attribute's value is not present in the referenced attribute value. e.g.; Insertion of a student with BRANCH_CODE 'ME' in STUDENT relation will result in an error because 'ME' is not present in BRANCH_CODE of BRANCH.

Deletion/ Updation Anomaly in Referenced Relation:

We can't delete or update a row from REFERENCED RELATION if the value of REFERENCED ATTRIBUTE is used in the value of REFERENCING ATTRIBUTE. e.g; if we try to delete a tuple from

BRANCH having BRANCH_CODE 'CS', it will result in an error because 'CS' is referenced by BRANCH_CODE of STUDENT, but if we try to delete the row from BRANCH with BRANCH_CODE CV, it will be deleted as the value is not been used by referencing relation. It can be handled by the following method:

On Delete Cascade

It will delete the tuples from REFERENCING RELATION if the value used by REFERENCING ATTRIBUTE is deleted from REFERENCED RELATION. e.g.; For, if we delete a row from BRANCH with BRANCH_CODE 'CS', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be deleted.

On Update Cascade

It will update the REFERENCING ATTRIBUTE in REFERENCING RELATION if the attribute value used by REFERENCING ATTRIBUTE is updated in REFERENCED RELATION. e.g;, if we update a row from BRANCH with BRANCH_CODE 'CS' to 'CSE', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be updated with BRANCH_CODE 'CSE'.

Super Keys

Any set of attributes that allows us to identify unique rows (tuples) in a given relationship is known as super keys. Out of these super keys, we can always choose a proper subset among these that can be used as a primary key. Such keys are known as Candidate keys. If there is a combination of two or more attributes that are being used as the primary key then we call it a Composite key.

Codd Rules in Relational Model

Edgar F Codd proposed the relational database model where he stated rules. Now these are known as Codd's Rules. For any database to be the perfect one, it has to follow the rules.

Advantages of the Relational Model

- Simple model: Relational Model is simple and easy to use in comparison to other languages.
- Flexible: Relational Model is more flexible than any other relational model present.
- Secure: Relational Model is more secure than any other relational model.
- Data Accuracy: Data is more accurate in the relational data model.
- Data Integrity: The integrity of the data is maintained in the relational model.
- Operations can be Applied Easily: It is better to perform operations in the relational model.

Disadvantages of the Relational Model

- Relational Database Model is not very good for large databases.
- Sometimes, it becomes difficult to find the relation between tables.
- Because of the complex structure, the response time for queries is high.

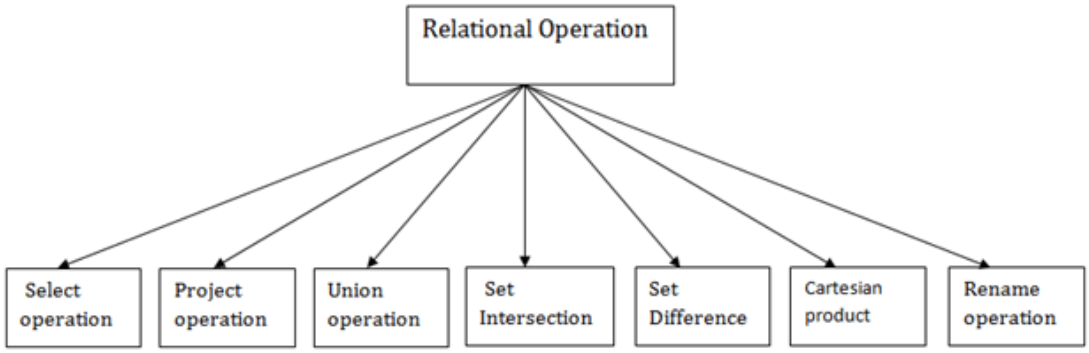
Characteristics of the Relational Model

- Data is represented in rows and columns called relations.
- Data is stored in tables having relationships between them called the Relational model.
- The relational model supports the operations like Data definition, Data manipulation, and Transaction management.
- Each column has a distinct name and they are representing attributes.
- Each row represents a single entity.

2. Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

Types of Relational operation



1. Select Operation:

The select operation selects tuples that satisfy a given predicate.

It is denoted by sigma (σ).

Notation: σ p(r)

Where:

σ is used for selection prediction

r is used for relation

p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

1. σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:
   - This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
   - It is denoted by ∏.

1. Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

1. ∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:
   - Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
   - It eliminates the duplicate tuples. It is denoted by ∪.

1. Notation: R ∪ S

A union operation must hold the following condition:
   - R and S must have the attribute of the same number.
   - Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:

1. ∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

1. Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product
   - The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
   - It is denoted by X.

1. Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

1. EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

1. ρ(STUDENT1, STUDENT)

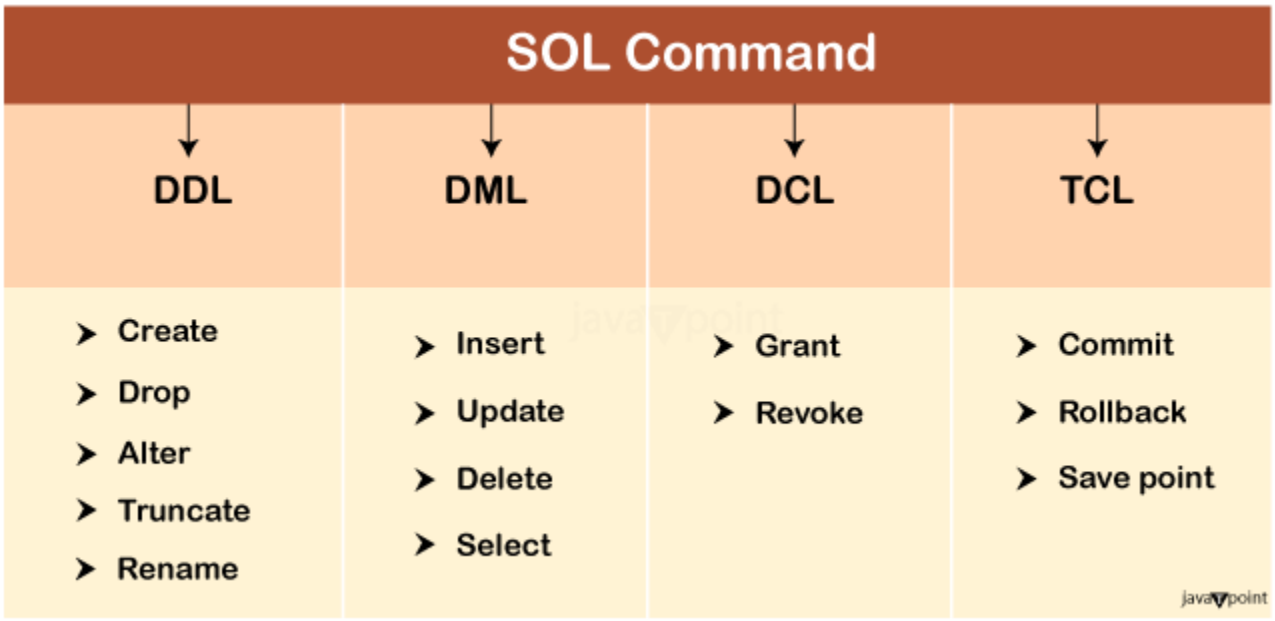
3. Basic SQL Queries

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.
- 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

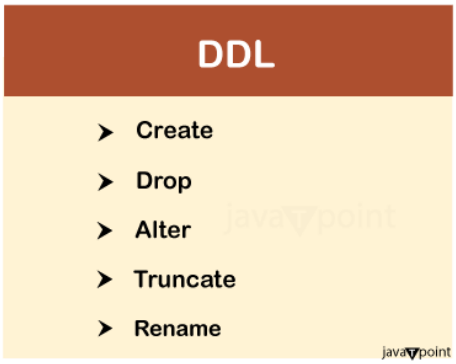
There are four types of SQL commands: DDL, DML, DCL, TCL.



1. 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.

Following are the some commands that come under DDL:



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

Syntax:

1. CREATE TABLE TABLE_NAME (COLUMN_NAMES DATATYPES [...]);

   In above statement, TABLE_NAME is the name of the table, COLUMN_NAMES is the name of the columns and DATATYPES is used to define the type of data.

Example:

1. CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

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

Syntax: To DROP a table permanently from memory

1. DROP TABLE table_name [cascade constraint];

The cascade constraint is an optional parameter which is used for tables which have foreign keys that reference the table being dropped. If cascade constraint is not specified and used attempt to drop a table that has records in a child table, then an error will occur. So by using cascade constraints, all child table foreign keys are dropped.

Example

1. DROP TABLE EMPLOYEE;

c. 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.

Following are the list of modifications that can be done using ALTER command.

- With the use of ALTER commands we can add or drop one or more columns form existing tables.
- Increase or decrease the existing column width by changing the data type
- Make an existing mandatory column to optional.
- Enable or disable the integrity constraints in a table. We can also add, modify or delete the integrity constraints from a table.
- We can also specify a default value for existing column in a table.

Adding new columns in Table:

With the use of ALTER table command we can add new columns existing table.

Syntax: To add a new column in the table

1. ALTER TABLE table_name ADD column_name column-definition;

In the above syntax, where table_name corresponds to the name of the table, column-definition corresponds to the valid specifications for a column name and data type.

EXAMPLE:

1. ALTER TABLE STU_DETAILS ADD (ADHAR_NUM VARCHAR2 (15));

Syntax: To ADD a multiple column from a table.

ALTER TABLE table_name ADD column_name1, column_name2;

Example:

1. ALTER TABLE STU_DETAILS ADD ADHAR_NUM, NAME;

Adding constraints in a Table:

You can also add constraints to an existing table. For example: If you forget to add a primary key constraint to a table during creation, you can add it using the ALTER TABLE statement.

Syntax: To ADD a constraint from a table.

1. ALTER TABLE table_name ADD (column_name column-definition CONSTRAINT constraint_name);

Example:

1. ALTER TABLE STU_DETAILS ADD (CONSTRAINT PK_STU_DETAILS PRIMARY KEY (STU_ID);

Following points should be kept in mind while adding new columns/relationships to existing tables.

- No need for parentheses if you add only one column or constraints.

- You can add a column at any time if NULL is not specified. You can add a new column with NOT NULL if the table is empty.

Modifying Column using ALTER:

With the use of ALTER table we can modify column and constraint in the existing table. These statements can increase or decrease the column widths and changing a column from mandatory to optional.

Syntax:

1. ALTER TABLE table_name MODIFY (column definitions....);

Example:

1. ALTER TABLE STU_DETAILS MODIFY (ADHAR_NUM VARCHAR2 (20));

SQL does not allow column widths to be reduced even if all column values are of valid length. So the values should be set to NULL to reduce the width of the columns. It is also not possible to reduce the width of the ADHAR_NUM column from 18 to 12 even if all values in the ADHAR_NUM column are less than 12 characters, unless all al values in the name column are null. You can modify the column form NULL to NOTNULL constraints if there is no record in that column in the table.

Example:

1. ALTER TABLE STU_DETAILS MODIFY (ADHAR_NUM VARCHAR2 (20) NOT NULL);

Drop column and constraints using ALTER

You cannot only modify columns but you can also drop them entirely if it is no longer required in a table. Using drop statement in alter command we can also remove the constraints form the table.

Syntax: To drop a column from a table.

1. ALTER TABLE table_name DROP COLUMN column_name;

Example:

Example:

1. ALTER TABLE STU_DETAILS MODIFY (ADHAR_NUM VARCHAR2 (20));

SQL does not allow column widths to be reduced even if all column values are of valid length. So the values should be set to NULL to reduce the width of the columns. It is also not possible to reduce the width of the ADHAR_NUM column from 18 to 12 even if all values in the ADHAR_NUM column are less than 12 characters, unless all al values in the name column are null. You can modify the column form NULL to NOTNULL constraints if there is no record in that column in the table.

Example:

1. ALTER TABLE STU_DETAILS MODIFY (ADHAR_NUM VARCHAR2 (20) NOT NULL);

Drop column and constraints using ALTER

You cannot only modify columns but you can also drop them entirely if it is no longer required in a table. Using drop statement in alter command we can also remove the constraints form the table.

Syntax: To drop a column from a table.

1. ALTER TABLE table_name DROP COLUMN column_name;

Example:

1. ALTER TABLE STU_DETAILS DROP PRIMARY KEY CASCADE;

- You can also enable or disable the key constraint in a table. It can be done in various situations such as: when loading large amount of data into table, performing batch operations, migrating the organizations legacy data.

Example: To disable constraint

1. ALTER TABLE STU_DETAILS DISABLE CONSTRAINT FK_STU_DETAILS;

Example: To Enable constraint

1. ALTER TABLE STU_DETAILS ENABLE CONSTRAINT FK_STU_DETAILS;

- Instead of dropping a column in a table, we can also make the column unused and drop it later on. It makes the response time faster. After a column has been marked as unused, the column and all its contents are no longer available and cannot be recovered in the future. The unused columns will not be retrieved using Select statement

Example:

1. ALTER TABLE STU_DETAILS SET UNUSED COLUMN ADHAR_NUM;

RENAMING TABLE

SQL provides the facility to change the name of the table by using a ALTER TABLE statement.

Syntax:

1. ALTER TABLE <OLD_TABLENAME> Rename to <NEW_TABLENAME>;

Example:

1. ALTER TABLE STU_NAME Rename to STUDENT_NAME;

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

Syntax:

1. TRUNCATE TABLE table_name;

Example:

1. TRUNCATE TABLE EMPLOYEE;

e. Rename: It is used to rename the table.

Syntax:

1. Rename <OLD_TABLENAME> to <NEW_TABLENAME>;

In the above syntax, Rename is a command, <OLD_TABLENAME> is the name of the table and <NEW_TABLENAME> is the name that you have changed.

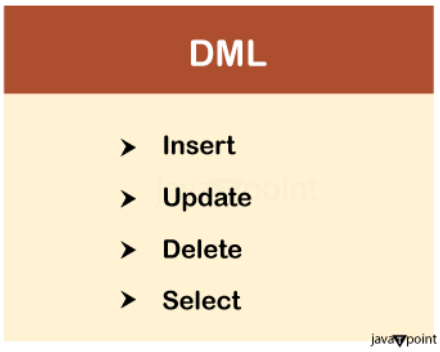
Example:

1. Rename STU_NAME to STUDENT_NAME;

2. 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.

Following are the some commands that come under DML:



a. INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table. To insert a new row into a table you must be your on schema or INSERT privilege on the table.

Following are the list of points should be considered while inserting the data into tables.

- SQL uses all the columns by default if you do not specify the column name while inserting a row.
- The number of columns in the list of column name must match the number of values that appear in parenthesis after the word "values".
- The data type for a column and its corresponding value must match.

Syntax: To add row in a table

1. INSERT INTO TABLE_NAME
2. (col1, col2, col3,.... col N)
3. VALUES (value1, value2, value3, ... valueN);

Or

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

In the above syntax, TABLE_NAME is the name of the table in which the data will be inserted. The (col1, col2, col3, col N) are optional and name of the columns in which values will be inserted. The value1 corresponds to the value of be inserted in col1 and similarly value2 corresponds to the value of be inserted in col2 and so on.

For example:

1. INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");

Syntax: To add multiple rows in a table

1. INSERT INTO TABLE_NAME
2. (col1, col2, col3,.... col N)
3. VALUES (value1, value2, value3, ... valueN), (value1, value2, value3, ... valueN);

For example:

1. INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS"), ("Raman", "DBMS"), ("Priya", " DBMS");

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

Syntax: To update record in a table

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

In the above syntax, table_name is the name of the table, the column_name is the name of column in the table to be modified, and value1 corresponds to the valid SQL values. The "WHERE" is a condition that restricts the rows updated for which the specified condition is true. If condition is not

specified is not defined then SQL updates all the rows in the table. It contains comparison and logical operators etc.

The following the list of points should be remembered while executing the UPDATE statement.

- It references only one table.
- In the SET clause atleast one column must be assigned an expression for the update statement,
- In the where clause you could also give multiple conditions for update statement.

For example:

```
1. UPDATE students
2. SET User_Name = 'Sonoo'
3. WHERE Student_Id = '3'
```

c. DELETE: It is used to remove one or more row from a table. To delete rows from the table, it must be in your schema or you must have delete privilege.

Syntax: To Delete a record from table

```
1. DELETE FROM table_name [WHERE condition];
```

In the above syntax, condition is used in the where clause to filter the records that are actually being removed. You can remove zero or more rows from a table. If you do not use where condition then DELETE statement will remove all the rows from the table. You can also use one or multiple conditions in WHERE clause.

For example:

```
1. DELETE FROM javatpoint
2. WHERE Author="Sonoo";
```

d. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax: It is used for retrieve the records from table

```
1. SELECT expressions
2. FROM TABLES
3. WHERE conditions;
```

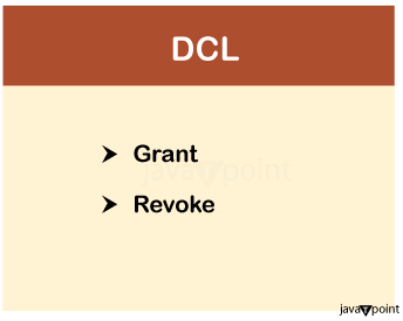
For example:

```
1. SELECT emp_name
2. FROM employee
3. WHERE age > 20;
```

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Following are the some commands that come under DCL:



Syntax:

1. GRANT <obj_priv> ON <obj_name> To <username>;

In the above syntax, obj_priv> is the DML statement like Insert, Delete , update and Select and <obj_name> is a table, view etc. and username is the name of the authorized user.

Example

1. GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

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

Syntax:

1. REVOKE <obj_priv> ON <obj_name> FROM <username>;

In the above syntax, obj_priv> is the DML statement like Insert, Delete , update and Select and <obj_name> is a table, view etc. and username is the name of the user from whom the permission is revoked.

Example

1. REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

4. Transaction Control Language

Transactions are atomic i.e. either every statement succeeds or none of statement succeeds. There are number of Transaction Control statements available that allow us to control this behavior. These statements ensure data consistency. 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.

Following are the some commands that come under TCL:

a. Commit: Commit command is used to save all the transactions to the database. It makes your changes permanent and ends the transaction.

Syntax: To permanently save the changes

1. COMMIT;

Example:

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. COMMIT;

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database. Rollback also serves to end the current transaction and begin a new one.

Consider a Situation where you have completed a series of INSERT, UPDATE or DELETE statements but have not yet explicitly committed them and yiu encounter a problem such as computer failure, then SQL will automatically rollback any uncommitted work.

Syntax: To remove the changes

1. ROLLBACK;

Example:

```
1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. ROLLBACK;
```

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

Syntax:

```
1. SAVEPOINT SAVEPOINT_NAME;
```

In the above syntax, SAVEPOINT_NAME is the name given to savepoint.

To selectively ROLLBACK a group of statements within a large transaction use the following command is used.

```
1. Rollback TO <save_point_name>
```

Example:

```
1. DELETE FROM CUSTOMERS WHERE AGE = 15;
2. SAVEPOINT A;
3. DELETE FROM CUSTOMERS WHERE AGE = 35;
4. ROLLBCAK TO A;
```