Why Normalization:-

- Normalization is process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & delete anomaly.

4.1.1 Anomalies in DBMS :-

- There are three types of anomalies that occur when the database is not normalized.
- These are – **Insertion, Update and Deletion anomaly.**

  Example,

  Manufacturing company stores the employee details in table :-

EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_DEPT
101	Richa	Delhi	D001
101	Richa	Delhi	D002
123	Megha	Agra	D890
166	Ganesh	Chennai	D900
166	Ganesh	Chennai	D004

- The above table is not normalized. Now we will see the problem that we face when a table is not normalized.

➢ **Update anomaly :-**

- An update anomaly, is a data inconsistency that results from data redundancy and a partial update.
- In the above table **we have two rows for employee Richa** as she belongs to two departments of the company.

- If we want to update the address of Richa then we have to update the same in rows or the data will become inconsistent.
- If somehow, the correct address gets updated in one department but not in other then as per the database, Richa would be having two different addresses, which is not correct and would lead to inconsistent data.

➢ **Insert anomaly :-**

- An insertion anomaly is the inability to add data to the database due to absence of other data.
- Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

➢ **Delete anomaly :-**

- A deletion anomaly is the unintended loss of data due to deletion of other data.
- Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having EMP_DEPT as D890 would also delete the information of employee Megha Since she is assigned only to this department.

<u>**To Overcome these anomalies we need to normalize the data.**</u>

Normalization Rules :-

Dependency

- Dependencies in DBMS is a relation between two or more attributes.

It has the following types in DBMS :-

1. **Functional Dependency**
2. **Fully-Functional Dependency**
3. **Transitive Dependency**
4. **Multivalued Dependency**
5. **Partial Dependency**

Prime Attribute and Non - Prime Attribute :-

- Attributes that form a candidate key of a relation, i.e attributes of candidate key, are called **Prime attributes.** And rest of the attributes of relation are **Non prime.**

  For Example,

  - For the relation {Roll_No, Name, City, Phone_No}
  - {Roll_No} is a candidate key and hence Roll_No is a prime attribute and Name, City, Phone_No Non-Prime attributes.

1. **Functional Dependency :-**

- If the information stored in a table can uniquely, determine another information in the same table, then it is called **Functional Dependency.**
- Consider it as an association between two attributes of the same relation.
- If P funcationally determines Q, then P -> Q.

For Example,

EMP_ID	EMP_NAME	EMP_AGE
E01	Amit	28
E02	Rohit	31

- In the above table, EMP_NAME is functionally dependent on EMP_ID because EMP_NAME can take only one value for the given value of EMP_ID.
- EMP_ID -> EMP_NAME

The same is displayed below,

EMP_ID -> EMP_NAME

Employee Name(EMP_NAME) is funcationally dependent on Employee ID(EMP_ID)

2. **Fully - Functional Dependency :-**

- An attribute is fully functional dependent on another attribute, if it is Functionally Dependent on that attribute and not on any of its proper subset.

For Example,

- An attribute Q is fully functional dependent on another attribute P, if it is Functionally Dependent on P and not on any of the proper subset of P.

For Example,

Table : Project_Cost

PROJECT_ID	PROJECT_COST
001	1000
002	5000

Table : Employee_Project

EMP_ID	PROJECT_ID	DAYS(SPENT ON THE PROJECT)
E009	001	320
E056	002	190

The above relations states :

- EMP_ID, PROJECT_ID, PROJECT_COST -> DAYS
- However, it is not fully functional dependent.
- Whereas the subset { EMP_ID, PROJECT_ID}can easily determine the {DAYS} spent on the project by the employee.
- This summarized and gives our fully functional dependency.
- { EMP_ID, PROJECT_ID} -> (DAYS)

3. **Partial Dependency : -**

   Partial Dependency occurs when a nonprime attribute is functionally dependent on part of a candidate key.

For Example,

Table : Student_Project

STUDENT_ID	PROJECT_NO	STUDENT_NAME	PROJECT_NAME
S01	199	Kalpesh	Geo Location
S02	120	Smita	Custer Exploration

- The prime key attributes are STUDENT_ID and PROJECT_NO.
- As stated, the non-prime attributes i.e. STUDENT_NAME and PROJECT_NAME should be functionally dependent on part of a candidate key, to be Partial Dependent.
- The STUDENT_NAME can be determined by STUDENT_ID that makes the relation Partial Dependent.
- The PROJECT_NAME can be determined by PROJECT_ID that makes the relation Partial Dependent.

4. **Transitive Dependency : -**

   By its nature, a transitive dependency requires three or more attributes (Or database columns) that have a functional dependency between them, meaning that Column A in table relies on Column B through an intermediate Column C.

For Example,

Table : Author

AUTHOR_ID	AUTHOR	BOOK	AUTHOR_CITY
AUTH_001	Bhushan Trivedi	Networking	Ahmedabad
AUTH_002	Bhushan Trivedi	Artificial Intelligent	Ahmedabad
AUTH_003	Prashant Dolia	Database Concepts	Surat

In the AUTHORS example above :

BOOK - > AUTHOR :

- Here, the book attribute determine the Author attribute.
- If you know the book name, you can learn the Author's name.
- However, just because we know the author's name Bhushan Trivedi, we still don't know the book name.

AUTHOR - > AUTHOR_CITY :

- Likewise, the Author attribute determines the Author_City, but not the other way around; just because we know the city does not mean we can determine the author.

➢ **But this table introduces transitive dependency :**

BOOK -> AUTHOR_CITY :

If we know the book name, we can determine the City via Author column.

5. Multi-Valued Dependency : -

- When existence of one or more rows in a table implies one or more other rows in the same table, then the multi-valued dependencies occur.
- If a table has attributes P.Q and R, Then Q and R are multi-valued facts of P.
- It is represented by **double arrow ->->**
- **For Example,**

  P ->->Q

  P ->->R
- In the above case, Multivalued Dependency exists only if Q and R are independent attributes.

  For Example.

- **Table : Student**

STUDENT_NAME	**COURSE_DISPLINE**	**ACTIVITIES**
Amit	Mathematics	Singing
Amit	Mathematics	Dancing
Yuvraj	Computers	Cricket
Akash	Literature	Dancing
Akash	Literature	Cricket
Akash	Literature	Singing

Armstrong Axioms OR Inference Rule (IR):

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR_1)

- In the reflexive rule, if Y is a subset of X, then X determines Y.
- If $X \supseteq Y$ then $X \rightarrow Y$

Example:

X = {a, b, c, d, e}
Y = {a, b, c}

2. Augmentation Rule (IR_2)

- The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.
- If $X \rightarrow Y$ then $XZ \rightarrow YZ$

Example:

For R(ABCD), **if** $A \rightarrow B$ then $AC \rightarrow BC$

3. Transitive Rule (IR_3)

- In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.
- If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

4. Union Rule (IR_4)

- Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.
- If X → Y and X → Z then X → YZ

Proof:

1. X → Y (given)
2. X → Z (given)
3. X → XY (using IR_2 on 1 by augmentation with X. Where XX = X)
4. XY → YZ (using IR_2 on 2 by augmentation with Y)
5. X → YZ (using IR_3 on 3 and 4)

5. Decomposition Rule (IR_5)

- Decomposition rule is also known as project rule. It is the reverse of union rule.
- This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.
- If X → YZ then X → Y and X → Z

Proof:

1. X → YZ (given)
2. YZ → Y (using IR_1 Rule)
3. X → Y (using IR_3 on 1 and 2)

6. Pseudo transitive Rule (IR_6)

- In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.
- If X → Y and YZ → W then XZ → W

Proof:

1. X → Y (given)
2. WY → Z (given)
3. WX → WY (using IR_2 on 1 by augmenting with W)
4. WX → Z (using IR_3 on 3 and 2)

Normalization :-

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
- It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

What is Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Data modification anomalies can be categorized into three types:

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.

- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Here are the most commonly used normal forms :

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

Normal Form	**Description**
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
4NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5NF	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

➢ **First Normal Form(1NF) :-**

As per the rule of first normal form, attribute(Column) of a table cannot hold multiple values. It should hold only atomic values.

Example,

Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this :

EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_MOBILE
101	Hershal	New Delhi	8912312390
102	Jigar	Kanpur	8812121212 9900012222
103	Roma	Chennai	7778881212
104	Leena	Bangalore	9990000123 8123450987

- Two employees(Jigar & Leena) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

- This table is not in 1NF as the rule says "each attribute of a table must have atomic(single) values", the EMP_MOBILE values for employees Jigar & Leena violates that rule.

To make the table complies with 1NF we should have the data like this :

EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_MOBILE
101	Hershal	New Delhi	8912312390
102	Jigar	Kanpur	8812121212
102	Jigar	Kanpur	9900012222
103	Roma	Chennai	7778881212
104	Leena	Bangalore	9990000123
104	Leena	Bangalore	8123450987

➢ **Second Normal Form(2NF) :-**

- A table is said to be in 2NF if both the following conditions hold :
- Table is in 1NF(First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.
- (i.e. There is no partial dependency)
- An attribute that is not part of any candidate key is known as non-prime attribute.

Example,

Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

TEACHER_ID	SUBJECT	TEACHER_AGE
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate keys: {TEACHER_ID, SUBJECT}

Non-prime attributes : TEACHER_AGE

- The table is in 1NF because each attribute has atomic values.
- However, it is not in 2NF because nonprime attribute TEACHER_AGE is dependent on TEACHER_ID alone which is a proper subset of candidate key.
- This violates the rule for 2NF as the rule says,"no non-prime attribute is dependent on the proper subset of any candidate key of the table."

To make the table 2NF we can break it in two tables like this :

TABLE : TEACHER_DETAILS :-

TEACHER_ID	TEACHER_AGE
111	38
222	38
333	40

TABLE : TEACHER_SUBJECT :-

TEACHER_ID	SUBJECT
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form(2NF).

➢ Third Normal Form (3NF) :-

- A table design is said to be in 3NF if both the following conditions hold :
- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should removed.
- An attribute that is not part of any candidate key is known as non-prime attribute.
- In other words 3NF can be explained like this : A table is in 3NF if it is in 2NF and for each functional dependency X -> Y at least one of the following conditions hold :
- X is super key of table.
- Y is a prime attribute of table

Example,

Suppose a company wants to store the complete address of each employee, they create a table named EMPLOYEE_DETAILS that look like this :

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY	EMP_DISTRICT
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Rahul	282007	TN	Chennai	Urrapakkan
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys : {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID,EMP_NAME,EMP_ZIP}… so on
Candidate Keys : {EMP_ID}

- Non-prime attributes : a;; attributes except emp_id are non-prime as they are not part of any candidate keys.
- Here, EMP_STATE, EMP_CITY & EMP_DISTRICT dependent on EMP_ZIP. And, EMP_ZIP is dependent on EMP_ID that makes non-prime attributes (EMP_STATE, EMP_CITY & EMP_DISTRICT) transitively dependent on super key(EMP_ID). This violates the rule of 3NF.
- EMP_IDàEMP_ZIP
- EMP_ZIPàEMP_STATE, EMP_ZIPàEMP_CITY, EMP_ZIPàEMP_DISTRICT
- EMP_IDàEMP_STATE, EMP_IDàEMP_CITY, EMP_IDàEMP_DISTRICT

To make this table complies with 3NF we have to break the table into table to remove the transitive dependency.

TABLE : EMPLOYEE_TABLE

EMP_ID	**EMP_NAME**	**EMP_ZIP**
1001	John	282005
1002	Ajeet	222008
1006	Rahul	282007
1101	Lilly	292008
1201	Steve	222999

TABLE : EMPLOYEE_ZIP

EMP_ZIP	EMP_STATE	EMP_CITY	EMP_DISTRICT
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkan
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

- **Boyce Codd Normal Form (BCNF) :-**

It is an advance version of 3NF that's why it is also referred as 3NF. BCNF is stricter that 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X -> Y, X should be the super key of the table.

Example,

Suppose there is a company where in employees work in more than one department, They store the data like this :

EMP_ID	EMP_NATIONALITY	EMP_DEPT	DEPT_TYPE	DEPT_NO_OF_EMP
1001	Austrian	Production and Planning	D001	200
1001	Austrian	Stores	D001	250
1002	American	Design and Technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above :

- EMP_ID - > EMP_NATIONALITY
- EMP_DEPT -> {DEPT_TYPE, DEPT_NO_OF_EMP}
- Candidate key : {EMP_ID, EMP_DEPT}
- The table is not in BCNF as neither EMP_ID nor EMP_DEPT alone are keys.

To make the table comply with BCNF we can break the table in three tables like this :

TABLE : EMP_NATIONALITY

EMP_ID	**EMP_NATIONALITY**
1001	Austrian
1001	Austrian
1002	American
1002	American

TABLE : EMP_DEPT TABLE

EMP_DEPT	**DEPT_TYPE**	**DEPT_NO_OF_EMP**
Production and Planning	D001	200
Stores	D001	250
Design and Technical support	D134	100
Purchasing department	D134	600

TABLE : EMP_DEPT_MAPPING

EMP_ID	**EMP_DEPT**
1001	Production and Planning
1001	Stores
1002	Design and Technical support
1002	Purchasing department

- **Functional dependencies :**

  EMP_ID -> EMP_NATIONALITY

  EMP_DEPT -> {DEPT_TYPE, DEPT_NO_OF_EMP}

- **Candidate keys :**
  - ✓ For first table : EMP_ID
  - ✓ For second table : EMP_DEPT
  - ✓ For third table : {EMP_ID, EMP_DEPT}

- This is now in BCNF as in both the functional dependencies left side part is a key.

Concepts of Structure Query Language (SQL) :-

What is 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

Data Types :-

- Each column in a database table is required to have a name and a data type.
- An SQL developer must decide what type of data that will be stored inside each column when creating a table.
- The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

SQL data types :-

- Numeric Types
- Alphanumeric Types
- Date – Time

Numeric Types :-

Name	Data type	No. of Bytes	Signed	Range
BOOLEAN	boolean	1	No	0 or 1
TINYINT	integer	1	No	0 to 255
SMALLINT	integer	2	Yes	-215 to 215-1
INTEGER	integer	4	Yes	-231 to 231-1
BIGINT	integer	8	Yes	-263 to 263-1
NUMERIC	number	no limit	Yes	(Max Scale, Max Precision) Max Scale = unlimited Max Precision = e(+/-)231
DECIMAL	decimal	no limit	Yes	(Max Scale, Max Precision) Max Scale = unlimited Max Precision = e(+/-)231
REAL	real	4	Yes	2 – 1074 to (2-2-52)*21023
FLOAT	float	4	Yes	2 – 1074 to (2-2-52)*21023
DOUBLE	double	4	Yes	2 – 1074 to (2-2-52)*21023

Alphanumeric Types :-

Name	Data type	Max Length	Description
Char	text(fix)	2GB for 32 bit OS	Stores exactly the number of characters specified for shorter strings. Accepts any UTF 8 Character.
Varchar	Text	2GB for 32 bit OS	Stores up to the specified number of characters. No padding (Same as long var char).
VARCHAR_IGNORECASE	Text	2GB for 32 bit OS	Stores up to the specified number of characters. Comparison are not case sensitive but stores capitals as you type them.
LONGVARCHAR	Text	2GB for 32 bit OS	Stores up to the max number of character indicated by user. It accepts any UTF 8 character.

Date-Time Types :-

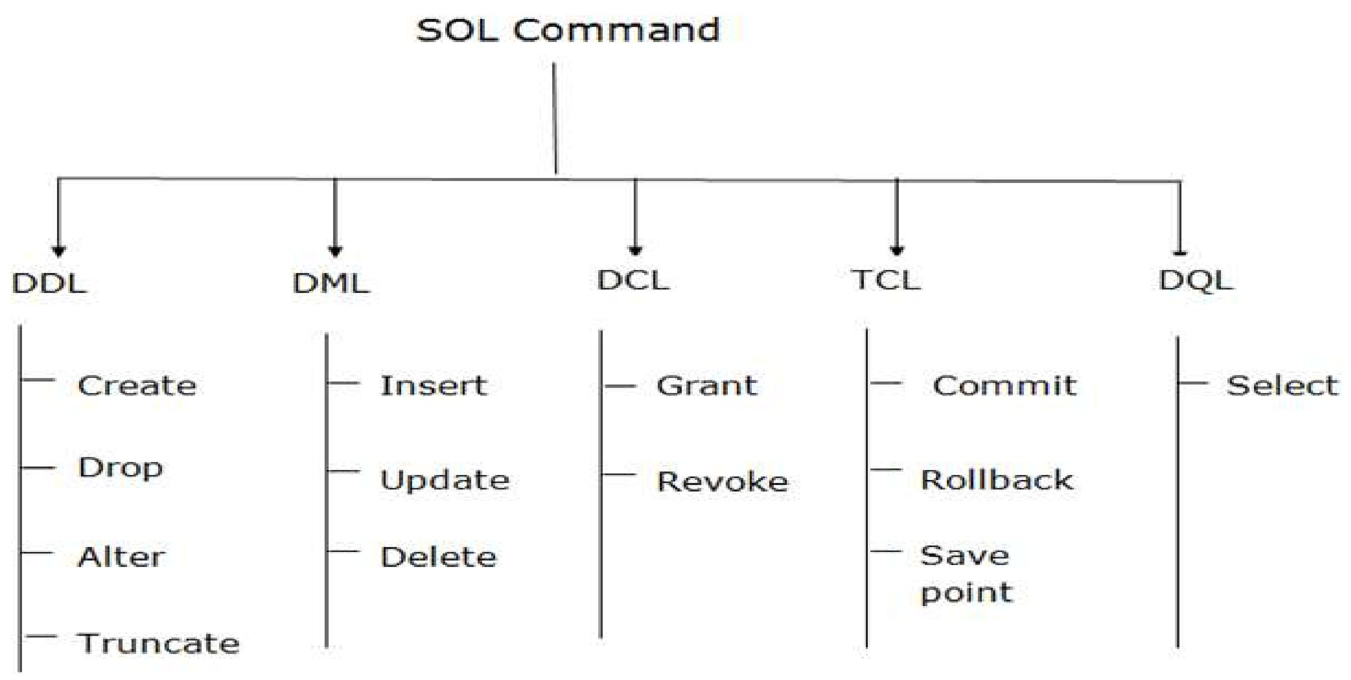
Name	**Description**	**Format**
Date	Stores month, day and year information	1/1/99 to 1/1/9999
Time	Stores hour, minute and second info	Seconds since 1/1/1970
Timestamp	Stores date and time information.	Timestamp

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 five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



4.4.1 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.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

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);
```

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

Syntax :

```
DROP TABLE table_name;
```

Example :

```
DROP TABLE EMPLOYEE;
```

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 table_name MODIFY(column_definitions....);
```

-: To Rename table :-

```
ALTER TABLE old_table_name RENAME TO new_table_name;
```

EXAMPLES :-

```
ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));
ALTER TABLE Student RENAME TO Student_Master;
```

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 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.

4.5.1 Here are some commands that come under DML :

- INSERT
- UPDATE
- DELETE

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 javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");
```

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 = valueN] [WHERE CONDITIO
N]
```

For example:

```
UPDATE students
SET User_Name = 'Sonoo'
WHERE Student_Id = '3'
```

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 javatpoint
WHERE Author="Sonoo";
```

1. Data Control Language

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

Here are some commands that come under DCL:

- Grant
- Revoke

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

Example

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

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

Example

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

2. Transaction Control Language

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:

- COMMIT
- ROLLBACK
- SAVEPOINT

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

Syntax:

```
COMMIT;
```

Example:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
COMMIT;
```

b. 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;
```

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

Syntax:

```
SAVEPOINT SAVEPOINT_NAME;
```

3. Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

a. 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:

```
SELECT expressions
FROM TABLES
WHERE conditions;
```

For example:

```
SELECT emp_name
FROM employee
WHERE age > 20;
```