UNIT-1: Introduction of Relational Model

1.1 Codd's Rules: (Dr. Edger Frank Codd)

- Dr. E. F. Codd developed a relational data model in 1970. In 1985, Dr. Codd published a list of 12 rules called as 12 codds that define an ideal relational database and has provided a guideline for the design of all relational database systems.
- The term guideline is used as no commercial database system fully confirms all 12 rules.

Rule 1: Information rule:

- All data should be presented in table form.
- Rows and columns have to be strictly unordered.

Rno	Name	Gender	City	Percentage
101	Raj	Male	Surat	88
102	Umang	Male	Anand	60
106	Raghav	Male	Surat	85

Rule 2: The guaranteed access rule:

- All data must be accessible without ambiguity. This can be accomplished through a combination of the table name, Primary Key and column name.
- Any single data can be addressed using logical addresses.
- These logical addresses are Table name, Primary key (Row) and Attribute (Column).

Rule 3: Systematic treatment of null values:

- A field should be allowed to remain empty. This involves the support of a null value, which is different from an empty string or number with a value of zero.
- This can't apply to the primary key. In addition, most database implementations support the concept of a NOT NULL field constraint that prevents null values in a specific table column.
- NULL value must be data type independent.
- NULL means
    - Not applicable
    - Missing data
    - No value

Rule 4: Dynamic online catalogue based on relational model:

- Data dictionary (Catalogue) must have description of data.
- The system must support online, inline, relational catalogue which is accessible by the authorized users using Query language. (Same like normal query on table.)
- A relational database must provide access to its structure through the same tools those are used to access the data. This is usually accomplished by storing the structure definition within special system tables.

Rule 5: Comprehensive Data Sub-Language Rule:

- The database must support at least one clearly defined language that includes functionality for data definition (CREATE, ALTER, DROP, and TRUNCATE), data manipulation (Insert, Update, Delete), data integrity (Constraints like Primary key, foreign key) and database transaction control (Commit, Rollback, Savepoint).
- All commercial relational databases use forms of standard SQL (i.e. Structure Query Language) as their supported comprehensive language

Rule 6: View updating rule:

- Data can be presented in different logical combinations called views. View is a virtual table which is generated based on a query. Each view should support the same full range of data manipulation that has direct access to a table available.
- In practice, providing update and delete access to logical views is difficult and is not fully supported by any current database.

Rule 7: High-Level Insert, Update, and Delete Rule

- A database must support high-level insertion, updation and deletion.
- It means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables.
- This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.
- Set operations like union, intersection and minus must also be supported.

Rule 8: Physical data Independency:

- Physical level changes must not affect the application system.
- Any modification is applied at the physical level, it doesn't affect the logical level and view level.
- Ex: If the file is moved from one drive to another drive then it must not affect the logical level and view level of the database.

Rule 9: Logical data independency:

- If any changes are applied to the logical level, it must not affect the view level of the database.
- For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application
- It is more difficult to achieve than physical level independency.

Rule 10: Integrity independency:

- Database must provide facility to specify constraints (Ex: NOT NULL, CHECK, DEFAULT etc.) without help of frontend program.
- The constraints related information must be stored in the Data Dictionary.
- Whenever any constraints need to modify or add then these must be allowed and can be applied using database systems only. (Without affecting the frontend program, constraints can be modified in the background within RDBMS).This rule makes RDBMS independent from frontend.

Rule 11: Distribution independency:

- Distribution of database within network should not be visible to the users (Clients)
- RDBMS must be distribution independent so data can be easily accessed by the user without bothering its storage. It helps in the Distributed database.

Rule 12: The non-subversion rule:

- If a low level interface is allowed to the user then also the user can't bypass the integrity constraints.
- Users can work from any level, but integrity constraints must be followed by the user before storing data within the system.
- Strict security is provided to the database.

1.2 Relational operations Algebra

- Relational algebra is the procedural query language.
- It takes instances of relations as input and produces instances of relations as output.
- Many operations are performed to get the output.
- Example: (Student table)

Rno	Name	Gender	City	Percentage
101	Raj	Male	Surat	88
102	Umang	Male	Anand	60
103	Meera	Female	Bharuch	76
104	Maya	Female	Surat	80
106	Raghav	Male	Surat	85

1.2.1 Select (σ)

- Select operation selects the tuples (rows) which satisfy the given conditions.
- Syntax:

$\sigma_p(r)$

- Where **σ** stands for selection predicate and **r** stands for relation. ***p*** is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like – =, ≠, ≥, < , >, ≤.

- Example:

1. $\sigma_{Gender="Male"}$ (tblstudent)

Rno	Name	Gender	City	Percentage
101	Raj	Male	Surat	88
102	Umang	Male	Anand	60
106	Raghav	Male	Surat	85

2. $\sigma_{Gender="Male" \text{ and } City="Surat"}$ (tblstudent)

Rno	Name	Gender	City	Percentage
101	Raj	Male	Surat	88
106	Raghav	Male	Surat	85

3. $\sigma_{Percentage > 80}$ (tblstudent)

Rno	Name	Gender	City	Percentage
101	Raj	Male	Surat	88
106	Raghav	Male	Surat	85

1.2.2 Project (π)

- Projection allows to display only specific attributes' (fields') data in place of all.
- Syntax:

$\Pi_{attributes}$ (relation)

- Example:

1. $\Pi_{Rno, Name}$ (tblstudent)

Rno	Name

101	Raj
102	Umang
103	Meera
104	Maya
106	Raghav

2. $\Pi_{Rno,\ Name}\ (\sigma_{Gender="Male"\ and\ City="Surat"}\ (tblstudent))$

Rno	Name
101	Raj
106	Raghav

1.2.3 Union (U)

- If A and B are the two tables (relations) and if we apply UNION then it displays all the records of A and B.
- It removes duplicate records.
- Total numbers of attributes of A and B must be the same.
- Attribute domains must be compatible with each other.
- Syntax: A U B
- Example:

1. $\Pi_{Rno,\ Name}\ (\sigma_{Gender="Male"}\ (tblstudent))\ U\ \Pi_{Rno,\ Name}\ (\sigma_{City="Surat"}\ (tblstudent))$

Rno	Name
101	Raj
102	Umang
104	Maya
106	Raghav

1.2.4 Intersection (∩)

- If A and B are two tables (Relations) and if we apply INTERSECTION then it displays only those tuples which are available in both tables.
- It removes duplicate records.
- Total numbers of attributes of A and B must be the same.

- Attribute domains must be compatible with each other.
- Syntax: A ∩ B
- Example:

1. $\Pi_{Rno, Name}$ ($\sigma_{Gender="Male"}$ (tblstudent))∩ $\Pi_{Rno, Name}$ ($\sigma_{City="Surat"}$ (tblstudent))

Rno	Name
101	Raj
106	Raghav

1.2.5 rename (ρ)

- It is used to rename the output of the relation.
- Syntax: ρ_{name} (expression)
- Example:

1. $\rho_{stud\_info\ (sno, sname)}$ ($\Pi_{Rno, Name}$ (tblstudent))
   (Note: Here the name of the table is also changed.)

Sno	Sname
101	Raj
102	Umang
103	Meera
104	Maya
106	Raghav

2. $\rho_{(sno, sname}$($\Pi_{Rno, Name}$ (tblstudent))

Sno	Sname
101	Raj
102	Umang
103	Meera
104	Maya
106	Raghav

1.3 Transaction control language

- It includes the statements which are used to manage the transactions occurring within the database.
- It manages the changes made by DML statements.
- Because of TCL, the atomicity feature is fulfilled within the database.
- A transaction is a set of SQL statements which Oracle treats as a Single Unit means all the statements are executed successfully or none of the statements are executed.
- Permanent changes are applied to the transaction only after the commit statement.
- If the commit statement is not executed and any system failure occurs, then the temporary changes are not permanently saved within the database but the whole transaction is rollbacked.
- It includes following statements:

1.3.1 Commit

- It is used to save the data permanently after the completion of a transaction.
- To execute a COMMIT automatically whenever an INSERT, UPDATE or DELETE command is executed, you can set the AUTOCOMMIT environment variable as **SET AUTOCOMMIT ON.**
- Syntax: COMMIT;
- Example:

```
INSERT INTO tblstudent VALUES (105, 'Het', 'Male','Surat',89);
COMMIT;
(It permanently inserts the record within the database.)
```

1.3.2 Rollback

- It is used to restore the database to its original form at the last commit. It gives undo effect to the currently executed transaction.
- Syntax: ROLLBACK; OR ROLLBACK TO <SavepointName>;
- Example:

```
INSERT INTO tblstudent VALUES (107, 'Heta', 'Female','Surat',80);
INSERT INTO tblstudent VALUES (108, 'Hetvi', 'Female','Anand',60);
INSERT INTO tblstudent VALUES (109, 'Hervi', 'Female','Nadiad',70);
ROLLBACK;
(All these statements related to changes being removed from the temporary storage means 107,108,109 records' entries are not updated within the database.)
```

1.3.3 Savepoint

- It is used to divide the transaction into numbers of sections (parts) to save the transaction related data temporarily.
- This pointer is used by the ROLLBACK statement whenever required.

- Syntax: SAVEPOINT <SavePointName>;
- Example:

```
INSERT INTO tblstudent VALUES (107, 'Heta', 'Female','Surat',80);
SAVEPOINT S1;
INSERT INTO tblstudent VALUES (108, 'Hetvi', 'Female','Anand',60);
SAVEPOINT S2;
INSERT INTO tblstudent VALUES (109, 'Hervi', 'Female','Nadiad',70);
```

- Different options:
  - ROLLBACK;
    The whole transaction is rollbacked.
  - ROLLBACK TO S1;
    Transaction is rollbacked to the SAVEPOINT S1 place. Means 108 and 109 records related modifications are removed.
    Now we can COMMIT or can apply ROLLBACK from this point.
  - ROLLBACK TO S2;
    Transaction is rollbacked to the SAVEPOINT S2 place. Means 109 record related modifications are removed.
    Now we can COMMIT or can apply ROLLBACK from this point.
  - COMMIT;
    Transaction is permanently saved to the database.
    All SAVEPOINTs are removed.

1.4 Data Control language

- It includes the statements which are used to provide security to the database.
- With multiple users' access to the database, these commands are necessary to provide privileges to specific users regarding access to the database.
- As per the privileges (permissions), users can access specific data.
- It controls the access of the database.
- It includes following statements:

1.4.1 Grant

- It is used to give permission to the users (which means who can use which database) for accessing the database.
- Syntax:

```
GRANT privilege_name
ON object_name
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];
```

- Privilege_name: Access rights or privileges which are granted to the user.
  - SELECT: Allow accessing of SELECT statement for specific table.
  - INSERT: Allow inserting records within a specific table.

  - UPDATE: Allow to modify the records within a specific table.
  - DELETE: Allow to delete the records from the specific table.
  - REFERENCES: Allows to provide constraints to specific table.
  - ALTER: Allows to modify the structure of a specific table.
  - INDEX: Allows to create index on specific table.
- Object_name: Name of database object like TABLE, VIEW, SEQUENCE, STORED PROC.
- User_name: Name of the user to whom you want to grant permission.
- PUBLIC: Allow access rights to all users.
- Role_name: Set of privileges grouped together.
- WITH GRANT OPTION: Allows the user to grant access rights to other users.
- Examples:
  - GRANT SELECT ON student TO fy1;
    Allow username fy1 to select the data from the Student table. Only read-only permission is granted to fy1 for this table. So if fy1 wants to insert a record in the student table then it is not allowed to.
  - GRANT SELECT, INSERT, UPDATE, DELETE ON student TO fy1;
    Allow sy1 to perform insert, delete, update and select the data from the Student table.
  - GRANT ALL on student TO sy1;
    Allow all permissions to sy1 of the student table.
  - GRANT SELECT on student TO PUBLIC;
    Allow all users to fetch the data from the student table. Means the student table is read only to all users.

1.4.2 Revoke

- It is used to withdraw some or all of the granted privileges given using GRANT statement.
- Syntax:

```
REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}
```

- Examples:
  - REVOKE SELECT ON tblstudent FROM fy1;
    It revokes select permission from the student table from user fy1.
  - REVOKE ALL on tblstudent FROM sy1;
    It removes all rights given to sy1 for the student table.
  - REVOKE SELECT on tblstudent FROM PUBLIC;
    It removes selection rights from all users of the student table.