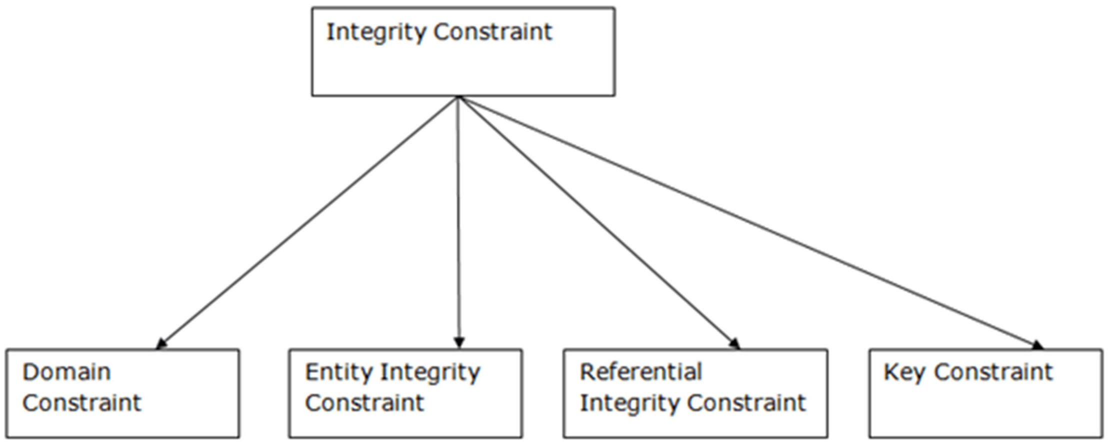
a) Integrity Constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

A Schema may contain zero or more integrity Constraints (an Assertion is just a special type of integrity Constraint: it is not necessarily dependent on a single Base table as simple Constraints are.) An SQL Constraint is a named rule which helps define valid sets of values by putting limits on the results of INSERT, UPDATE or DELETE operations performed on a Base table, an Assertion, by contrast, may define valid sets of values for individual rows of a Base table or for an entire Base table or it may define the set of valid values required to exist among a number of Base tables. Constraints are dependent on some Schema – the <Constraint name> must be unique within the Schema the Constraint belongs to – and are created and dropped using standard SQL statements.

There are four Constraint variations –
UNIQUE Constraints, PRIMARY KEY Constraints, FOREIGN KEY Constraints
and CHECK Constraints.

1. A UNIQUE Constraint defines one or more Columns of a Table as unique Columns: it is satisfied if no two rows in the Table have the same non-null values in the unique Columns.
2. A PRIMARY KEY Constraint is a UNIQUE Constraint that specifies PRIMARY KEY: it is satisfied if *(a)* no two rows in the Table have the same non-null values in the unique Columns and *(b)* none of the primary key Columns are NULL. UNIQUE Constraints and PRIMARY KEY Constraints describe a Base table's candidate keys.
3. A FOREIGN KEY Constraint defines one or more Columns of a Table as referencing Columns whose values must match the values of some corresponding referenced Columns in a referenced Base table (the referenced Columns must be UNIQUE Columns for the referenced Table). It is satisfied if, for every row in the referencing Table, the values of the referencing Columns are equal to those of the corresponding referenced Columns in some row of the referenced Table. (If either Table contains NULLs, satisfaction of the FOREIGN KEY Constraint depends on the Constraint's match type.) FOREIGN KEY Constraints describe linkages between Base tables.
4. A CHECK Constraint defines a search condition: it is violated if the result of the condition is FALSE for any row of the Table. An Assertion is a CHECK Constraint that may operate on multiple Tables.



i. Domain Constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

ID	Name	Semester	Age
1000	AAA	1	22
1001	BBB	3	23
1002	CCC	5	21
1003	DDD	6	A

Not allowed. Because AGE is an integer attribute

ii. Entity integrity constraints

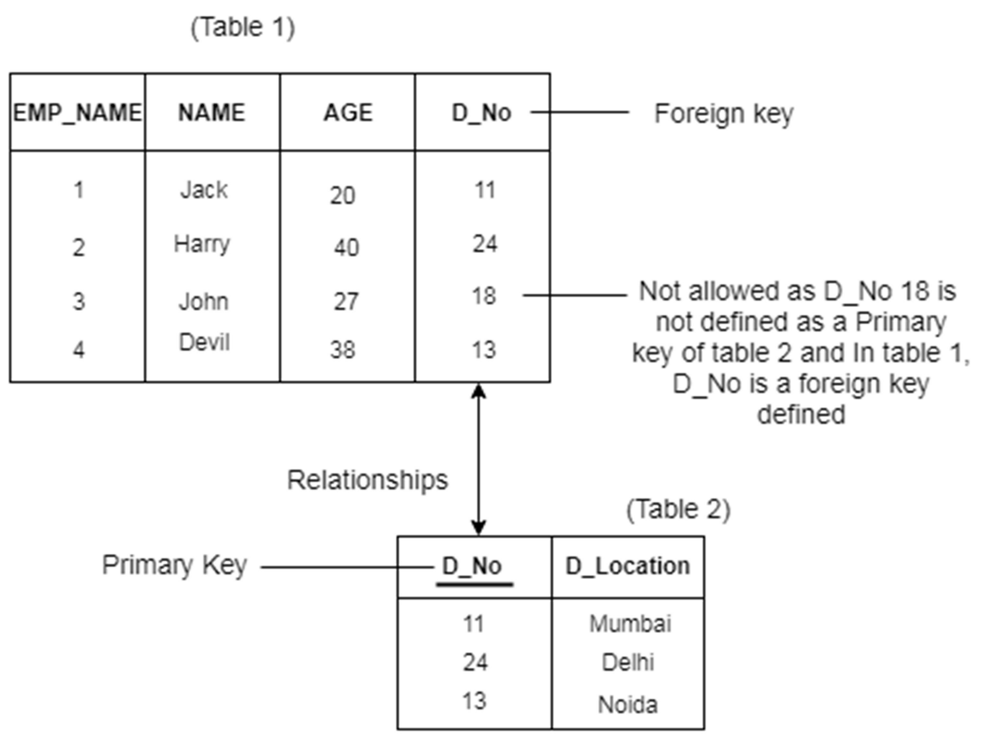
- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

ID	Name	Semester	Age
1000	AAA	1	22
1001	BBB	3	23
1002	CCC	5	21
	DDD	6	23

Not allowed as primary key can't contain a NULL value

iii. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.



(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

(Table 2)

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

Foreign key
Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined
Relationships
Primary Key

iv. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

ID	Name	Semester	Age
1000	AAA	1	22
1001	BBB	3	23
1002	CCC	5	21
1001	DDD	6	23

Not allowed. Because all row must be unique

b) Triggers and Assertions in SQL

What are Triggers?

A trigger is a database object that is associated with the table, it will be activated when a defined action is executed for the table. The CREATE TRIGGER statement allows you to create a new trigger that is fired automatically whenever an event such as INSERT, DELETE, or UPDATE occurs against a table.

The following illustrates the syntax of the CREATE TRIGGER statement:

```
CREATE TRIGGER [schema_name.]trigger_name
ON table_name
AFTER  {[INSERT],[UPDATE],[DELETE]}
[NOT FOR REPLICATION]
AS
{sql_statements}
```

In this syntax:

- The schema_name is the name of the schema to which the new trigger belongs. The schema name is optional.
- The trigger_name is the user-defined name for the new trigger.
- The table_name is the table to which the trigger applies.
- The event is listed in the AFTER clause. The event could be INSERT, UPDATE, or DELETE. A single trigger can fire in response to one or more actions against the table.
- The NOT FOR REPLICATION option instructs SQL Server not to fire the trigger when data modification is made as part of a replication process.
- The sql_statements is one or more Transact-SQL used to carry out actions once an event occurs.

“Virtual” tables for triggers: INSERTED and DELETED

SQL Server provides two virtual tables that are available specifically for triggers called INSERTED and DELETED tables. SQL Server uses these tables to capture the data of the modified row before and after the event occurs.

The following table shows the content of the INSERTED and DELETED tables before and after each event:

DML event	INSERTED table holds	DELETED table holds
INSERT	Rows to be inserted	Empty
UPDATE	New rows modified by the update	Existing rows modified by the update
DELETE	Empty	Rows to be deleted

Assertions

When a constraint involves 2 (or) more tables, the table constraint mechanism is sometimes hard and results may not come as expected. To cover such situation SQL supports the creation of assertions which are constraints not associated with only one table. And an assertion statement should ensure a certain condition will always exist in the database. DBMS always checks the assertion whenever modifications are done in the corresponding table.

Syntax :

```
CREATE ASSERTION  [ assertion_name ]

CHECK ( [ condition ] );
```

Example:

```
CREATE TABLE sailors (sid int,sname varchar(20), rating int,primary key(sid),

CHECK(rating >= 1 AND rating <=10)

CHECK((select count(s.sid) from sailors s) + (select count(b.bid)from boats b)<100) );
```

In the above example, we are enforcing CHECK constraint that the number of boats and sailors should be less than 100. So here we are able to CHECK constraints of two tablets simultaneously.

c) Security and Authorization in SQL

A DBMS system always has a separate system for security which is responsible for protecting database against accidental or intentional loss, destruction or misuse.

- **Security Levels:**
    - Database level: - DBMS system should ensure that the authorization restriction needs to be there on users.
    - Operating system Level: - Operating system should not allow unauthorized users to enter in system.
    - Network Level: - Database is at some remote place and it is accessed by users through the network so security is required.

- **Security Mechanisms:**
    - **Access Control(Authorization)**
        - Which identifies valid users who may have any access to the valid data in the Database and which may restrict the operations that the user may perform?
        - For Example The movie database might designate two roles: “users” (query the data only) and “designers”(add new data)user must be assigned to a role to have the access privileges given to that role.
        - Each applications is associated with a specified role. Each role has a list of authorized users who may execute/Design/administers the application.
    - **Authenticate the User:**
        - Which identify valid users who may have any access to the data in the Database?
        - Restrict each user’s view of the data in the database
        - This may be done with help of concept of views in Relational databases.
    - **Cryptographic control/Data Encryption:**
        - Encode data in a cryptic form(Coded)so that although data is captured by unintentional user still he can’t be able to decode the data.
        - Used for sensitive data, usually when transmitted over communications links but also may be used to prevent by passing the system to gain access to the data.

- **Inference control:**
    - Ensure that confidential information can’t be retrieved even by deduction.
    - Prevent disclosure of data through statistical summaries of confidential data.
- **Flow control or Physical Protection:**
    - Prevents the copying of information by unauthorized person
    - Computer systems must be physically secured against any unauthorized entry.
- **Virus control:**
    - At user level authorization should be done to avoid intruder attacks through humans.
    - There should be mechanism for providing protection against data virus.
- **User defined control:**
    - Define additional constraints or limitations on the use of database.
    - These allow developers or programmers to incorporate their own security procedures in addition to above security mechanism.

Authorization

- Authorization is finding out if the person,once identified,is permitted to have the resource.
- Authorization explains that what you can do and is handled through the DBMS unless external security procedures are available.
- Database management system allows DBA to give different access rights to the users as per their requirements.
- Basic Authorization we can use any one form or combination of the following basic forms of authorizations
    - Resource authorization:-Authorization to access any system resource. e.g. sharing of database, printer etc.
    - Alternation Authorization:- Authorization to add attributes or delete attributes from relations
    - Drop Authorization:-Authorization to drop a relation.

- **Granting of privileges:**
  - A system privilege is the right to perform a particular action,or to perform an action on any schema objects of a particular type.
  - An authorized user may pass on this authorization to other users.This process is called as granting of privileges..
  - Syntax:

```
o GRANT <privilege list>
o ON<relation name or view name>
o TO<user/role list>
```

Example:

The following grant statement grants user U1,U2 and U3 the select privilege on Emp_Salary relation:

```
GRANT select
ON Emp_Salary
TO U1,U2 and U3.
```

- **Revoking of privileges:**
  - We can reject the privileges given to particular user with help of revoke statement.
  - To revoke an authorization,we use the revoke statement.
  - Syntax:

```
o REVOKE <privilege list>
o ON<relation name or view name>
o FROM <user/role list>[restrict/cascade]
```

- Example:

The revocation of privileges from user or role may cause other user or roles also have to loose that privileges.This behavior is called cascading of the revoke.

```
Revoke select
ON  Emp_Salary
FROM U1,U2,U3.
```

- **Some other types of Privileges:**
  - **Reference privileges:**

    SQL permits a user to declare foreign keys while creating relations. Example: Allow user U1 to create relation that references key ‘Eid’ of Emp_Salary relation.

    ```
    GRANT REFERENCES (Eid)

    ON Emp_Salary

    TO U1.
    ```

  - **Execute privileges:**

    This privileges authorizes a user to execute a function or procedure. Thus,only user who has execute privilege on a function Create_Acc() can call function.

    ```
    GRANT EXECUTE

    ON Create_Acc

    TO U1.
    ```