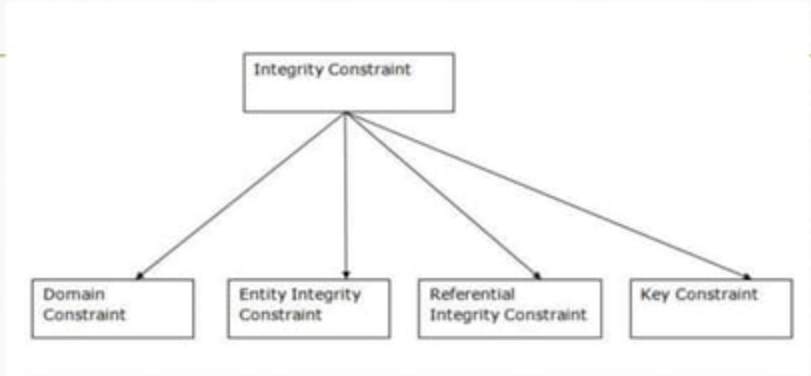
INTEGRITY CONSTRAINTS

Integrity Constraints

- Integrity constraints are used to ensure **accuracy and consistency** of the data in a relational database.
- Data integrity is handled in a relational database through the concept of referential integrity.
- 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.

Types of Integrity Constraints



Domain constraint

- **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.
- Domain constraint are :

  Not null

  Unique

  Check (<predicate>)

NOT NULL Constraint

- By default, a column can hold NULL values. The NOT NULL constraint enforces a column to NOT accept NULL values. This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

Example

```
CREATE TABLE STUDENT(
ROLL_NO  INT  NOT NULL ,
STU_NAME VARCHAR (35)  NOT NULL,
STU_AGE INT  NOT NULL,
EXAM_FEE INT DEFAULT 10000,
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO) )
```

- In the above example we have set the not null constraint on ROLL_NO column of STUDENT table. Now the roll no values can not be null .

UNIQUE constraint

- UNIQUE Constraint enforces a column or set of columns to have unique values.
- If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.

```
CREATE TABLE STUDENT
(ROLL_NO INT NOT NULL UNIQUE,
STU_NAME VARCHAR (35) NOT NULL ,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (35) PRIMARY KEY (ROLL_NO)
);
```

Now the roll number is set to be unique ,which means no two similar values of roll number is accepted .

CHECK constraint

- **The check clause :**This constraint is used for specifying range of values for a particular column of a table.
- When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range

```
CREATE TABLE STUDENT(
ROLL_NO  INT  CHECK(ROLL_NO >1000) ,
STU_NAME VARCHAR (35)  NOT NULL,
STU_AGE INT  NOT NULL,
EXAM_FEE INT DEFAULT 10000,
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO) );
```

In the above example we have set the check constraint on ROLL_NO column of STUDENT table. Now, the ROLL_NO field must have the value greater than 1000.

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.
- In the diagram ,say Emp_Id is primary key in the table. Thus from the definition of Entity Integrity, the value of Emp_Id cannot be null as it unique identifies an employee record in the table.
- Thus no primary key column of any row in a table can have a null value.

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

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

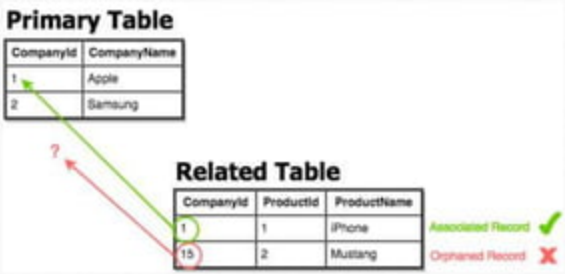
Referential integrity

- **Referential integrity** refers to the accuracy and consistency of data within a relationship. In relationships, data is linked between two or more tables.
- This is achieved by having the foreign key (in the associated table) reference a primary key value (in the primary – or parent – table).

- As a Relational Database Management System (RDBMS), **SQL Server** uses the referential integrity constraint to ensure that data in one table points to data in another table—and doesn't point to data that doesn't exist.
- **SQL Server** uses constraints, triggers, rules, and defaults to enforce **referential integrity**.

Example

- For example, if we delete record number 15 in a primary table, we need to be sure that there's no foreign key in any related table with the value of 15. We should only be able to delete a primary key if there are no associated records. Otherwise, we would end up with an **orphaned record.**


Primary Table
CompanyId
CompanyName
1
Apple
2
Samsung
?
Related Table
CompanyId
ProductId
ProductName
1
1
iPhone
Associated Record
15
2
Mustang
Orphaned Record

- So referential integrity will prevent users from:
- Adding records to a related table if there is no associated record in the primary table.
- Changing values in a primary table that result in orphaned records in a related table.
- Deleting records from a primary table if there are matching related records.