**Relational Model Concepts**

The relational model represents the database as a collection of relations. Informally, each relation resembles a table of values.

When a relation is thought of as a table of values, each row in the table represents a collection of related data values.

**Domain →**

A domain D is a set of atomic values. By atomic we mean that each value in the domain is indivisible as far as the formal relational model is concerned.

Example → 10 digit phone numbers, name of a person, CGPA on a scale of 1.0 - 10.0, Academic branch/ department codes like CSE, IT, MECH, AE etc.

A data type or format is also specified for each domain.

* A relation schema , denoted by , is made up of a relation name and a list of attributes,. Each attribute is the name of a role played by some domain in the relation schema . is called the domain of and is denoted by .

**Tuples** → A Relation can also be represented as a set of n-tuples.

A relation is a mathematical relation of degree on the domains which is a subset of the Cartesian product (denoted by x) of the domains that define

If we denote the total number of values, or cardinality, in a domain by (assuming that all domains are finite), the total number of tuples in the Cartesian product is,

**Characteristics of Relations**

1. **Ordering of Tuples in a Relation** → A relation is not sensitive to the ordering of tuples.
2. **Ordering of Values within a Tuple** → The order of attributes and their values is not important as long as the correspondence between attributes and values is maintained.
3. **Values and Nulls in a Tuple →** Each value in a tuple is an atomic value. A special value called NULL, is used when **value unknown**, **value exists but is not available**, or **attribute does not apply** to this tuple (also known as **value undefined**).

**Relational Model Constraints**

There are generally many restrictions or constraints on the actual values in a database state.

Constraints on databases can generally be divided into three main categories:

1. Constraints that are inherent in the data model. We call these inherent model-based constraints or implicit constraints.
2. Constraints that can be directly expressed in the schemas of the data model, typically by specifying them in the DDL. We call these schema-based constraints or explicit constraints.
   1. **Domain Constraints** → Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain dom(A).
   2. **Constraints on NULLs** → A constraint on attributes specifies whether NULL values are or are not permitted. For example, if every STUDENT tuple must have a valid, non-NULL value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL
   3. **Key Constraints** → Super Key (Uniqueness Constraint), Candidate Key (Minimal Superkey Constraint), Primary Key.
   4. **Entity Integrity Constraints** → entity integrity constraint states that no primary key value can be NULL. This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that we cannot identify some tuples.
   5. **Referential Integrity Constraints** → The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

We use foreign key in order to define referential integrity more formally.The conditions for a foreign key, given below, specify a referential integrity constraint between the two relation schemas R1 and R2.

A set of attributes FK in relation schema R1 is a foreign key of R1 that references relation R2 if it satisfies the following rules:

1. The attributes in FK have the same domain(s) as the primary key attributes PK of R2; the attributes FK are said to reference or refer to the relation R2.
2. A value of FK in a tuple t1 of the current state r1(R1) either occurs as a value of PK for some tuple t2 in the current state r2(R2) or is NULL. In the former case, we have t1[FK] = t2[PK], and we say that the tuple t1 references or refers to the tuple t2.

In this definition, R1 is called the referencing relation and R2 is the referenced relation. If these two conditions hold, a referential integrity constraint from R1 to R2 is said to hold.

1. Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs or in some other way. We call these application-based or semantic constraints or business rules.  
     
   Another class of general constraints, sometimes called semantic integrity constraints, are not part of the DDL and have to be specified and enforced in a different way.

Examples of such constraints are the salary of an employee should not exceed the salary of the employee’s supervisor and the maximum number of hours an employee can work on all projects per week is 56. Such constraints can be specified and enforced within the application programs that update the database.

**Dealing with constraint violations**

Whenever any operations that change the database state are applied (Insert, Delete and Update operations), the integrity constraints specified on the relational database schema should not be violated.

**Insert Operation** →

1. Domain Constraint Violation
2. Key Constraint Violation
3. Entity Integrity Violation
4. Referential Integrity Violation

Dealing with insertion constraint violation would be

1. Rejection
2. Correct the reason for rejecting the insertion.
3. DBMS could either ask the user to change the value leading to cascade back.

**Delete Operation →**

1. Referential integrity Violation

Dealing with deletion constraint violation would be

1. Restrict
2. Set Null or Set Default
3. Cascade Delete

**Update Operation** →

This can be handled using the above two.