Constraints on Relational database model

In modeling the design of the relational database we can put some restrictions like what values are allowed to be inserted in the relation, and what kind of modifications and deletions are allowed in the relation. These are the restrictions we impose on the relational database.

In models like Entity-Relationship models, we did not have such features. Database Constraints can be categorized into 3 main categories:

Constraints that are applied in the data model are called Implicit Constraints.

Constraints that are directly applied in the schemas of the data model, by specifying them in the DDL(Data Definition Language). These are called as Schema-Based Constraints or Explicit Constraints.

Constraints that cannot be directly applied in the schemas of the data model. We call these Application based or Semantic Constraints.

So here we are going to deal with Implicit constraints.

Relational Constraints

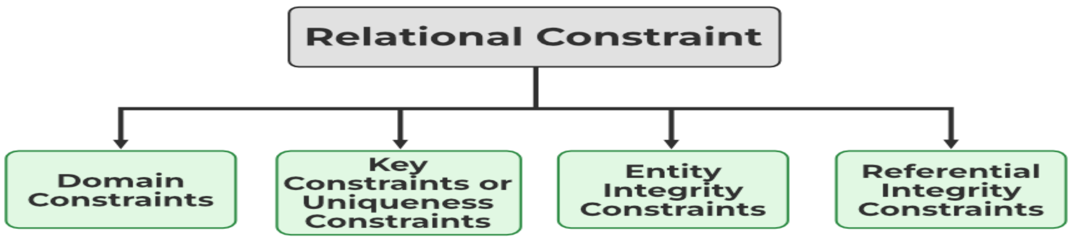
These are the restrictions or sets of rules imposed on the database contents. It validates the quality of the database. It validates the various operations like data insertion, updation, and other processes which have to be performed without affecting the integrity of the data. It protects us against threats/damages to the database. Mainly Constraints on the relational database are of 4 types

Domain constraints

Key constraints or Uniqueness Constraints

Entity Integrity constraints

Referential integrity constraints


Relational Constraint
Domain Constraints
Key Constraints or Uniqueness Constraints
Entity Integrity Constraints
Referential Integrity Constraints

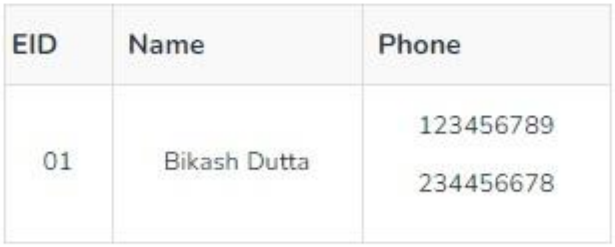
Let’s discuss each of the above constraints in detail.

1. Domain Constraints

Every domain must contain atomic values(smallest indivisible units) which means composite and multi-valued attributes are not allowed.

We perform a datatype check here, which means when we assign a data type to a column we limit the values that it can contain. Eg. If we assign the datatype of attribute age as int, we can’t give it values other than int datatype.

Example:



EID	Name	Phone
01	Bikash Dutta	123456789 234456678

Explanation: In the above relation, Name is a composite attribute and Phone is a multi-values attribute, so it is violating domain constraint.

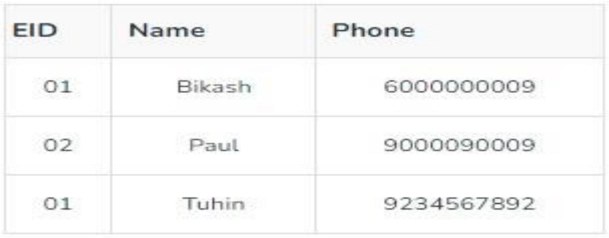
2. Key Constraints or Uniqueness Constraints

These are called uniqueness constraints since it ensures that every tuple in the relation should be unique.

A relation can have multiple keys or candidate keys(minimal superkey), out of which we choose one of the keys as the primary key, we don’t have any restriction on choosing the primary key out of candidate keys, but it is suggested to go with the candidate key with less number of attributes.

Null values are not allowed in the primary key, hence Not Null constraint is also part of the key constraint.

Example:



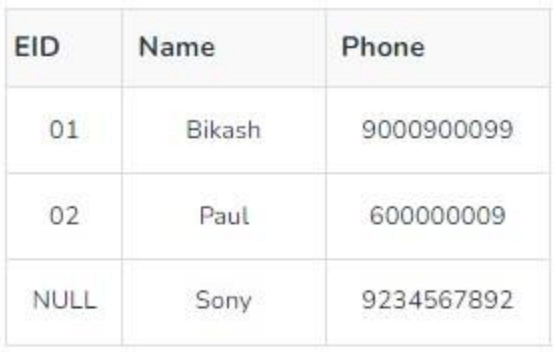
EID	Name	Phone
01	Bikash	6000000009
02	Paul	9000090009
01	Tuhin	9234567892

Explanation: In the above table, EID is the primary key, and the first and the last tuple have the same value in EID ie 01, so it is violating the key constraint.

3. Entity Integrity Constraints:

Entity Integrity constraints say that no primary key can take a NULL value, since using the primary key we identify each tuple uniquely in a relation.

Example:



EID	Name	Phone
01	Bikash	9000900099
02	Paul	600000009
NULL	Sony	9234567892

Explanation: In the above relation, EID is made the primary key, and the primary key can’t take NULL values but in the third tuple, the primary key is null, so it is violating Entity Integrity constraints.

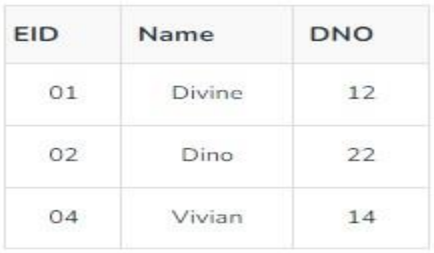
4. Referential Integrity Constraints

The Referential integrity constraint is specified between two relations or tables and used to maintain the consistency among the tuples in two relations.

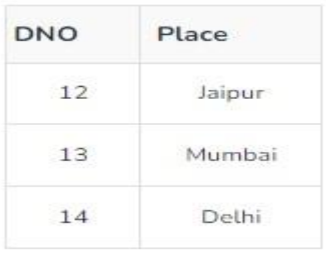
This constraint is enforced through a foreign key, when an attribute in the foreign key of relation R1 has the same domain(s) as the primary key of relation R2, then the foreign key of R1 is said to reference or refer to the primary key of relation R2.

The values of the foreign key in a tuple of relation R1 can either take the values of the primary key for some tuple in relation R2, or can take NULL values, but can’t be empty.

Example:



EID	Name	DNO
01	Divine	12
02	Dino	22
04	Vivian	14



DNO	Place
12	Jaipur
13	Mumbai
14	Delhi

Explanation: In the above tables, the DNO of Table 1 is the foreign key, and DNO in Table 2 is the primary key. DNO = 22 in the foreign key of Table 1 is not allowed because DNO = 22 is not defined in the primary key of table 2. Therefore, Referential integrity constraints are violated here.

Advantages of Relational Database Model

It is simpler than the hierarchical model and network model.

It is easy and simple to understand.

Its structure can be changed anytime upon requirement.

Data Integrity: The relational database model enforces data integrity through various constraints such as primary keys, foreign keys, and unique constraints. This ensures that the data in the database is accurate, consistent, and valid.

Flexibility: The relational database model is highly flexible and can handle a wide range of data types and structures. It also allows for easy modification and updating of the data without affecting other parts of the database.

Scalability: The relational database model can scale to handle large amounts of data by adding more tables, indexes, or partitions to the database. This allows for better performance and faster query response times.

Security: The relational database model provides robust security features to protect the data in the database. These include user authentication, authorization, and encryption of sensitive data.

Data consistency: The relational database model ensures that the data in the database is consistent across all tables. This means that if a change is made to one table, the corresponding changes will be made to all related tables.

Query Optimization: The relational database model provides a query optimizer that can analyze and optimize SQL queries to improve their performance. This allows for faster query response times and better scalability.

Disadvantages of the Relational Model

Few database relations have certain limits which can’t be expanded further.

It can be complex and it becomes hard to use.

Complexity: The relational model can be complex and difficult to understand, particularly for users who are not familiar with SQL and database design principles. This can make it challenging to set up and maintain a relational database.

Performance: The relational model can suffer from performance issues when dealing with large data sets or complex queries. In particular, joins between tables can be slow, and indexing strategies can be difficult to optimize.

Scalability: While the relational model is generally scalable, it can become difficult to manage as the database grows in size. Adding new tables or indexes can be time-consuming, and managing relationships between tables can become complex.

Cost: Relational databases can be expensive to license and maintain, particularly for large-scale deployments. Additionally, relational databases often require dedicated hardware and specialized software to run, which can add to the cost.

Limited flexibility: The relational model is designed to work with tables that have predefined structures and relationships. This can make it difficult to work with data that does not fit neatly into a table-based format, such as unstructured or semi-structured data.

Data redundancy: In some cases, the relational model can lead to data redundancy, where the same data is stored in multiple tables. This can lead to inefficiencies and can make it difficult to ensure data consistency across the database.