Integrity Rules and Constraints

In this lesson, we will discuss the different types of constraints present on relational databases.

Constraints are a very important feature in a relational model. In fact, the relational model supports the well-defined theory of constraints on attributes or tables. Constraints are useful because they allow a designer to specify the semantics of data in the database. Constraints are the rules that force DBMSs to check that data satisfies the semantics.

Relational integrity constraints

Relational integrity constraints are the conditions that must be present for a valid relation. These integrity constraints are derived from the rules in the mini-world that the database represents.

There are many types of integrity constraints, but we will focus on the following:

Domain constraints

Domain constraints specify that within each tuple, the value of each attribute must appear in the corresponding domain (in other words, it should belong to the appropriate data type). We have already discussed how domains can be specified in the previous lessons. The data types associated with domains typically include integers, real numbers, characters, booleans, etc. Let's consider the following example:

Std_Id	Name	Age
1	Jack	22
2	Jill	25
3	Ahmed	19
4	Anderson	A

Here, value ‘A’ is not allowed since only integer values can be taken by the `Age` attribute.

Entity integrity

To ensure entity integrity, it is required that every relation has a primary key. Neither the primary key nor any part of it can contain `NULL` values. This is because the presence of a `NULL` value in the primary key violates the uniqueness property. Let’s say we have the following table with `Std_Id` as the primary key:

Std_Id	Name	Age
1	Jack	22
2	Jill	25
3	Ahmed	19
NULL	Anderson	21

This relation does not satisfy the entity integrity constraint as here the primary key contains a `NULL` value.

Referential integrity constraint

This constraint is enforced when a foreign key references the primary key of a relation. It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be `NULL`.

Referential integrity constraint has two very important results:

Rule 1: We cannot insert a record into a referencing relation if the corresponding record does not exist in the referenced relation.

Example: Let us consider two relations: STUDENT and DEPARTMENT. Here, STUDENT is the referencing relation while DEPARTMENT is the referenced relation as `Dep_No` acts as the foreign key in the STUDENT relation.

Std_Id	Name	Dep_No
1	Jack	D1
2	Jill	D4
3	Ahmed	D5

Dep_No	Dep_Name
D1	Physics
D4	Biology
D5	Computer Science
D7	Electrical Engineering

Now we insert the following tuple into the STUDENT table; <4, ‘Anderson’, ‘D9’>

Std_Id	Name	Dep_No
1	Jack	D1
2	Jill	D4
3	Ahmed	D5
4	Anderson	D9

We now see that the STUDENT relation does not satisfy the referential integrity constraint. This is because, in the DEPARTMENT relation, no value of a primary key specifies department no. 9. Hence, the referential integrity constraint is violated.

Rule 2: We cannot delete or update the record of the referenced relation if the corresponding record exists in the referencing relation.

Example: Again consider the STUDENT and DEPARTMENT relations as seen above.

Now if we try to delete the tuple in the DEPARTMENT relation with `Dep_No` = 'D1'. This will again violate the referential integrity constraint, as some students are already enrolled in the course and this will lead to inconsistencies in the database.

Key constraints

As we know, a primary key uniquely identifies each record in a table. Therefore, it must have a unique value for each tuple in the relation *R*. For example:

Std_Id	Name	Age
1	Jack	22
2	Jill	25

3	Ahmed	19
1	Anderson	21

The key constraint is violated as the primary key `Std_Id` has the same value for the first and fourth tuple.

In the next lesson, we will define what a relational database schema is and how it is used to represent integrity constraints.