Unit-3

Domain Key Normal Form in DBMS

Prerequisites – Normal Forms, 4th and 5th Normal form, find the highest normal form of a relation

It is basically a process in database to organize data efficiently. Basically there are two goals of doing normalization these are as follows:

1. To remove repeated data or in simple words we can say to remove redundant data.
2. Second one is to ensure that there will be data dependencies.

Steps can be done to achieve Normalization:

1. Remove repeating groups or to eliminate repeating groups.
2. Eliminate or remove repeating data.
3. Remove those columns that are not dependent on Key.
4. Multiple relationship should be isolated independently.
5. Isolate Semantically Related Multiple Relationships

There are several types of normal forms, a lower numbered normal form always weaker than the higher numbered normal form. For example, 1st normal form is weaker than that of 2nd normal form. These are as: 1st, 2nd, 3rd, Boyce-code normal form, 4th, 5th, and domain key normal form. But, in this article, we will discuss only about Domain-Key Normal Form.

Domain key normal form (DKNF) –

There is no Hard and fast rule to define normal form up to 5NF. Historically the process of normalization and the process of discovering undesirable dependencies were carried through 5NF, but it has been possible to define the stricter normal form that takes into account additional type of dependencies and constraints.

The basic idea behind the *DKNF* is to specify the normal form that takes into account all the possible dependencies and constraints.

In simple words, we can say that DKNF is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.

In other words, a relation schema is said to be in DKNF only if all the constraints and dependencies that should hold on the valid relation state can be enforced simply by enforcing the domain constraints and the key constraints on the relation. For a relation in DKNF, it becomes very straight forward to enforce all the database constraints by simply checking that each attribute value is a tuple is of the appropriate domain and that every key constraint is enforced.

Reason to use DKNF are as follows:

1. To avoid general constraints in the database that are not clear key constraints.
2. Most database can easily test or check key constraints on attributes.

However, because of the difficulty of including complex constraints in a DKNF relation its practical utility is limited means that they are not in practical use, since it may be quite difficult to specify general integrity constraints.

Let's understand this by taking an example:

Example –

Consider relations CAR (MAKE, vin#) and MANUFACTURE (vin#, country),

Where vin# represents the vehicle identification number ‘country’ represents the name of the country where it is manufactured.
A general constraint may be of the following form:

If the MAKE is either ‘HONDA’ or ‘MARUTI’ then the first character of the vin# is a ‘B’ If the country of manufacture is ‘INDIA’
If the MAKE is ‘FORD’ or ‘ACCURA’, the second character of the vin# is a ‘B” if the country of manufacture is ‘INDIA’.

There is no simplified way to represent such constraints short of writing a procedure or general assertion to test them. Hence such a procedure needs to enforce an appropriate integrity constraint. However, transforming a higher normal form into domain/key normal form is not always a dependency-preserving transformation and these are not possible always.