Types of Constraints in DBMS or Relational Constraints in DBMS with Examples-

What are Database Constraints in DBMS ??

Database constraints are restrictions on the contents of the database or on database operations. It is a condition specified on a database schema that restricts the data to be inserted in an instance of the database.

Need of Constraints :

Constraints in the database provide a way to guarantee that :

- the values of individual columns are valid.
- in a table, rows have a valid primary key or unique key values.
- in a dependent table, rows have valid foreign key values that reference rows in a parent table.

Different Types of constraints in DBMS with Example :

- **Domain Constraints**
- **Tuple Uniqueness Constraints**
- **Key Constraints**
- **Single Value Constraints**
- **Integrity Rule 1 (Entity Integrity Rule or Constraint)**
- **Integrity Rule 2 (Referential Integrity Rule or Constraint)**
- **General Constraints**

Domain Constraints -

Domain Constraints specifies that what set of values an attribute can take. Value of each attribute X must be an atomic value from the domain of X. The data type associated with domains include integer, character, string, date, time, currency etc. An attribute value must be available in the corresponding domain. Consider the example below -

SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	1st	18
8003	Somvir	4th	22
8004	Sourabh	6th	A

Not Allowed. Because Age is an Integer Attribute.

Tuple Uniqueness Constraints -

A relation is defined as a set of tuples. All tuples or all rows in a relation must be unique or distinct. Suppose if in a relation, tuple uniqueness constraint is applied, then all the

rows of that table must be unique i.e. it does not contain the duplicate values. For example,

SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	2nd	18
8003	Somvir	4th	22
8004	Sourabh	6th	19

Not Allowed. Because all rows must be unique.

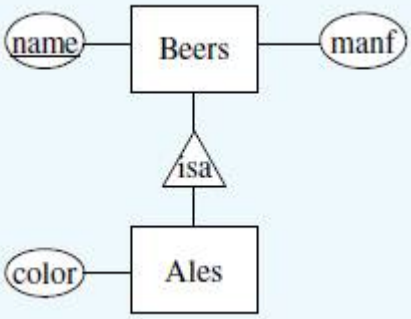
Key Constraints -

Keys are attributes or sets of attributes that uniquely identify an entity within its entity set. An Entity set E can have multiple keys out of which one key will be designated as the primary key. Primary Key must have unique and not null values in the relational table. In an subclass hierarchy, only the root entity set has a key or primary key and that primary key must serve as the key for all entities in the hierarchy. Example of Key Constraints in a simple relational table –

SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	1st	18
8003	Somvir	4th	22
8004	Sourabh	6th	45
8002	Tony	5th	23

Not allowed as Primary Key Values must be unique

Example of Key Constraints in an subclass hierarchy –



Single Value Constraints -

Single value constraints refers that each attribute of an entity set has a single value. If the value of an attribute is missing in a tuple, then we cal fill it with a "null" value. The null value for a attribute will specify that either the value is not known or the value is not applicable. Consider the below example-

SID	Name	Class (semester)	Age	Driving License Number
8001	Ankit	1st	19	DL-45698
8002	Srishti	2nd	18	DL-45871, DL-89740
8003	Somvir	4th	22	DL-95687
8004	Sourabh	6th	19	

Not allowed as a person does not have two driving licenses.

Allowed as a person may or may not have a driving license.

Integrity Rule 1 (Entity Integrity Rule or Constraint) -

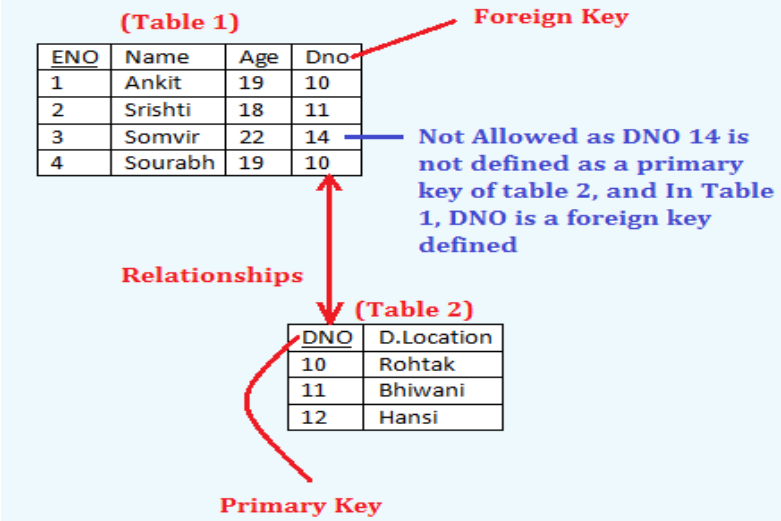
The Integrity Rule 1 is also called Entity Integrity Rule or Constraint. This rule states that no attribute of primary key will contain a null value. If a relation have a null value in the primary key attribute, then uniqueness property of the primary key cannot be maintained. Consider the example below-

SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	2nd	18
8003	Somvir	4th	22
	Sourabh	6th	19

Not allowed as primary key cannot contain a NULL value

Integrity Rule 2 (Referential Integrity Rule or Constraint) -

The integrity Rule 2 is also called the Referential Integrity Constraints. This rule states that if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2. For example,



Some more Features of Foreign Key -

Let the table in which the foreign key is defined is Foreign Table or details table i.e. Table 1 in above example and the table that defines the primary key and is referenced by the foreign key is master table or primary table i.e. Table 2 in above example. Then the following properties must be hold :

- Records cannot be **inserted** into a F**oreign table** if corresponding records in the master table do not exist.
- Records of the **master table or Primary Table** cannot be **deleted** or **updated** if corresponding records in the detail table actually exist.

General Constraints -

General constraints are the arbitrary constraints that should hold in the database. Domain Constraints, Key Constraints, Tuple Uniqueness Constraints, Single Value Constraints, Integrity Rule 1 (Entity Integrity) and 2 (Referential Integrity Constraints) are considered to be a fundamental part of the relational data model. However, sometimes it is necessary to specify more general constraints like the CHECK Constraints or the Range Constraints etc. **Check constraints** can ensure that only specific values are allowed in certain column. For example , if there is a need to allow only three values for the color like 'Bakers Chocolate', 'Glistening Grey' and 'Superior White', then we can apply the check constraint. All other values like 'GREEN' etc would yield an error. Range Constraints is implemented by BETWEEN and NOT BETWEEN. For example, if it is a requirement that student ages be within 16 to 35, then we can apply the range constraints for it. The below example will explain Check Constraint and Range Constraint –

CarID	Name	Model	Color
C-12378	Wagon-R	2008	Bakers Chocolate
C-23478	Wagon-R	2008	Glistening Grey
C-45823	Wagon-R	2004	Superior White
C-45874	Wagon-R	2009	Green

Not Allowed, as CHECK Constraint is applied.

SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	2nd	18
8003	Somvir	4th	22
NULL	Sourabh	6th	65

Not allowed, as the range defined is in between 16 and 35

Database Operations and Constraint Violations in DBMS - The Insert Operation

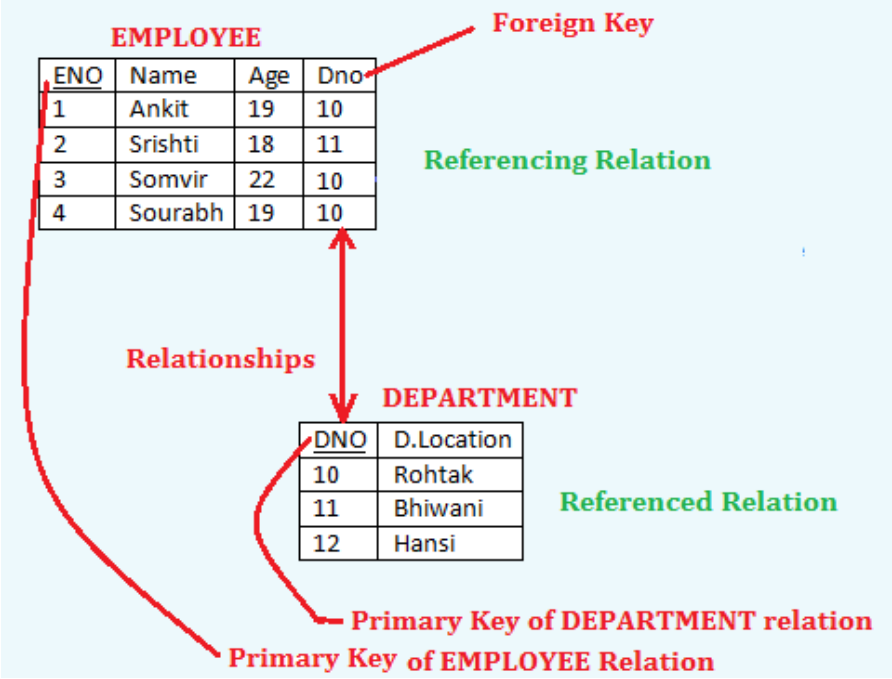
The basic Database operations performed on a relation are -

Insert Operation - The insert operation is used to insert a new tuple(s) on relations.

Delete Operation - The delete operation is used to delete the tuple(s).

Update or Modify Operation - The update operation is used to change/update the values of some attributes in existing tuples.

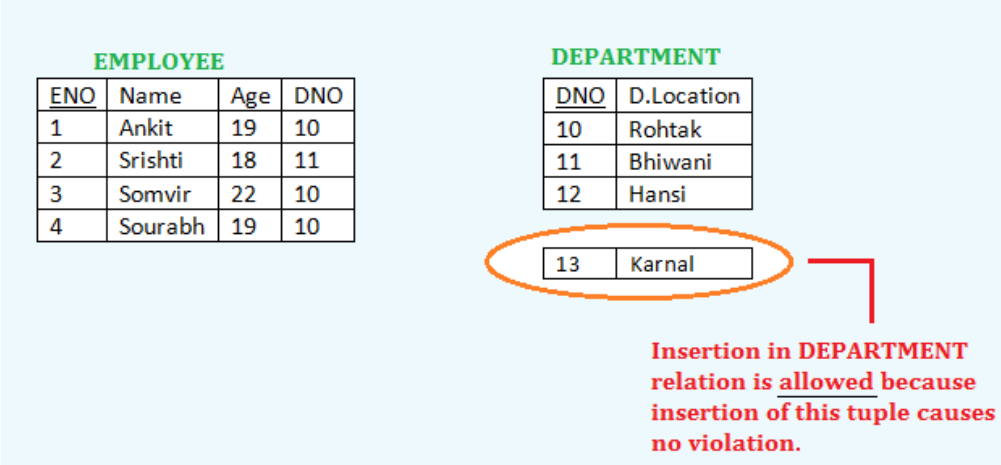
CLICK ON THE OPERATIONS TO KNOW MORE about Violations Whenever these above operations are performed, the integrity constraints should not be violated. Before describing operations, let us understand the following figure. Consider two existing relations named **EMPLOYEE** and **DEPARTMENT**.



Some Basic Points about the Figure -

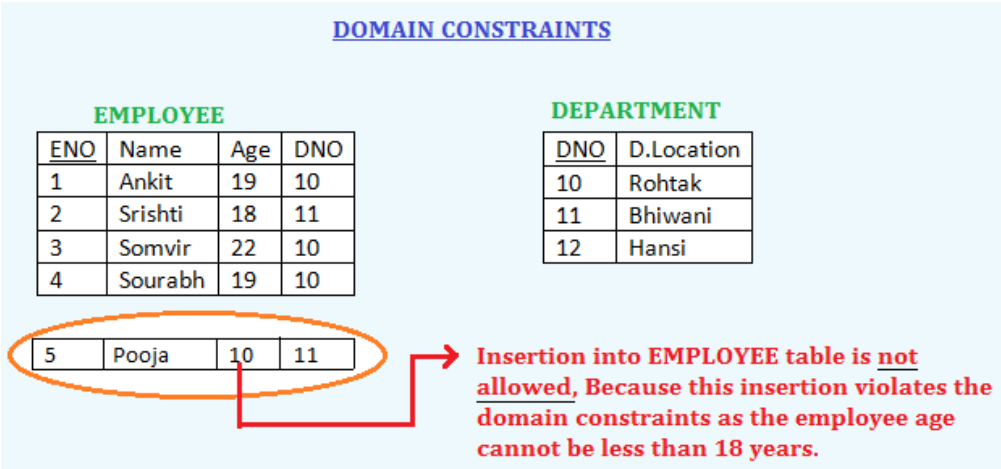
Here, ENO is a Primary Key and DNO is a Foreign Key in EMPLOYEE relation.
Table that contain candidate key is called **referenced relation** and
The table containing foreign key is called **referencing relation**.
So, the relation **DEPARTMENT is a referenced relation** and
The relation **EMPLOYEE is a referencing relation**.
The Insert Operation :

Insertion in a Referenced Relation (DEPARTMENT) - If we insert a tuple in the referenced relation , then it causes no violation. **For example,** Insertion of **<13, 'Karnal'>** in DEPARTMENT relation is allowed because insertion of this tuple causes no violation.

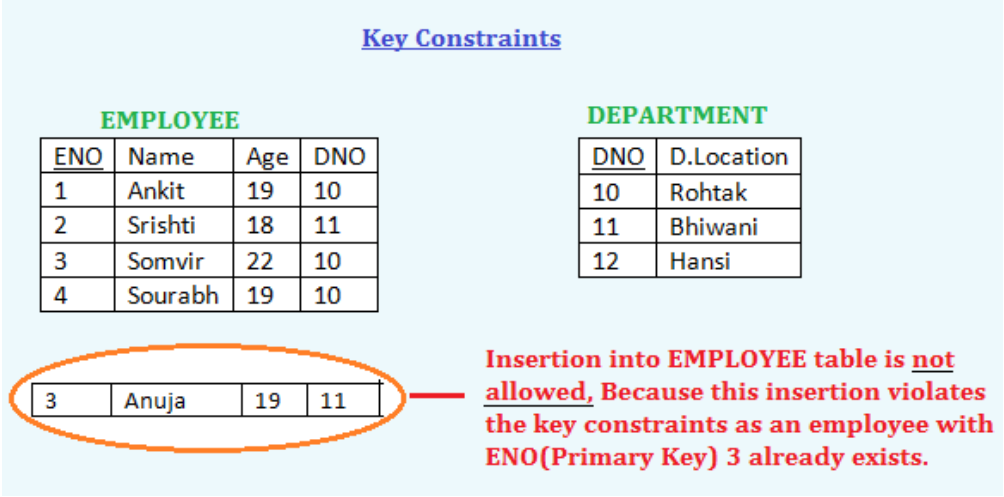


Insertion in a Referencing Relation (EMPLOYEE) - The insert operation in a referencing relation can violate any of four types of constraints -

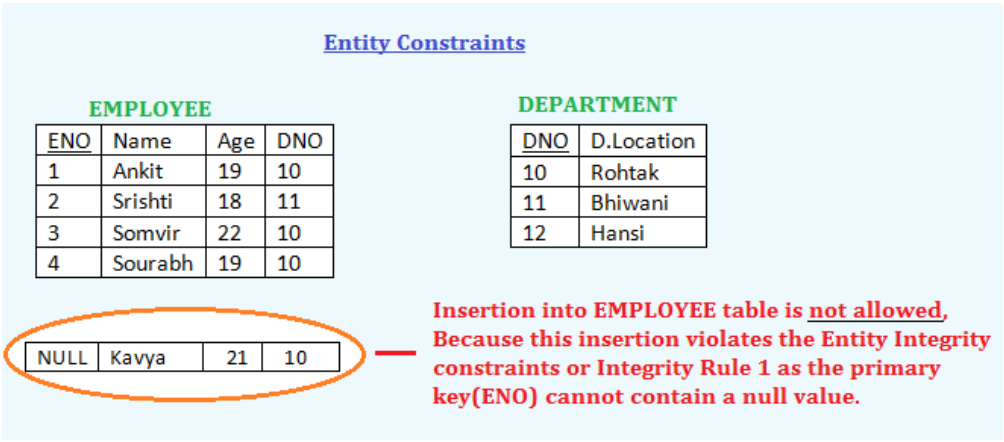
Domain Constraints - Insertion of the value **<5, 'Pooja', 10, 11>** into EMPLOYEE table is not allowed, Because this insertion violates the domain constraints as the employee age cannot be less than 18 years.



Key Constraints - Insertion of the value **<3, 'Anuja', 19, 11>** into EMPLOYEE table is not allowed, Because this insertion violates the key constraints as an employee with ENO(Primary Key) 3 already exists.



Entity Constraints - Insertion of the value **<NULL, 'Kavya', 21, 10>** into EMPLOYEE table is not allowed, Because this insertion violates the Entity Integrity constraints or Integrity Rule 1 as the primary key(ENO) cannot contain a null value.



Referential Integrity Constraints - Insertion of the value **<6, 'Ajit', 19, 16>** into EMPLOYEE table is not allowed, Because this insertion violates the Referential Integrity constraints or Integrity Rule 2 as there is no row or tuple with DNO=15 exists in DEPARTMENT relation.

Referential Integrity Constraints

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

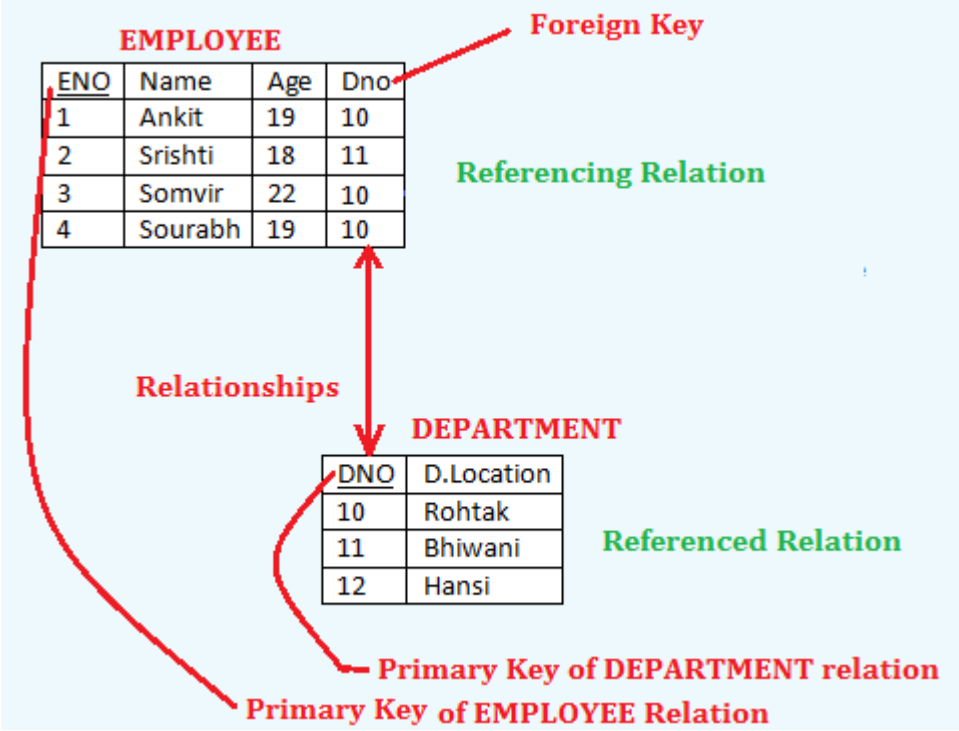
6	Ajit	19	16

Insertion into EMPLOYEE table is not allowed, Because this insertion violates the Referential Integrity constraints or Integrity Rule 2 as there is no row or tuple with DNO=15 exists in DEPARTMENT relation.

Database Operations and Dealing with Constraint Violations- The Delete Operation

The Delete Operation :

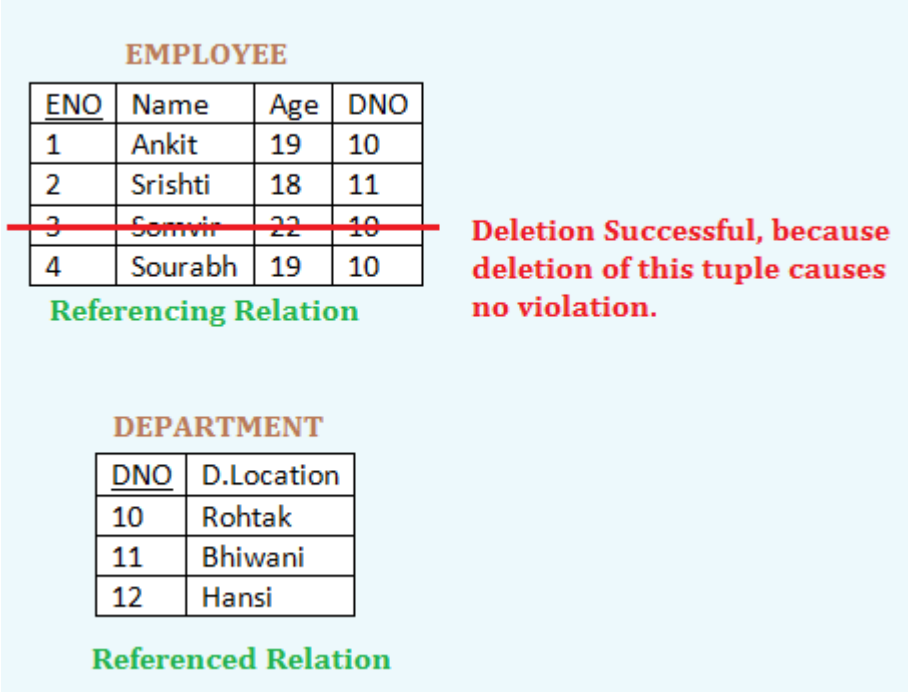
Consider two existing relations named **EMPLOYEE** and **DEPARTMENT**.



Some Basic Points about the Figure -

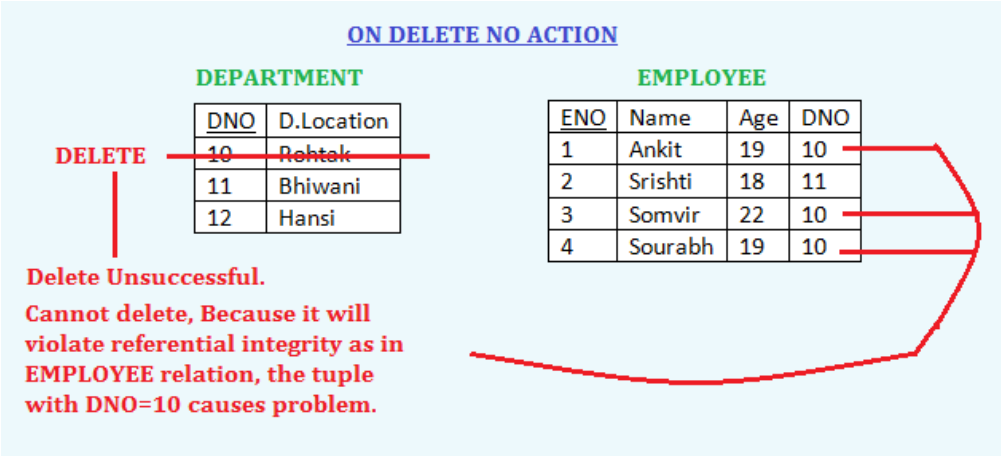
Here, ENO is a Primary Key and DNO is a Foreign Key in EMPLOYEE relation.
Table that contain candidate key is called **referenced relation** and
The table containing foreign key is called **referencing relation**.
So, the relation **DEPARTMENT is a referenced relation** and
The relation **EMPLOYEE is a referencing relation**.
The Delete Operation violates only referential Integrity Constraints or Integrity Rule 2.

Deletion in a Referencing Relation (EMPLOYEE relation) If we delete a tuple from the referencing relation, then it causes no violation. **For example,** Deletion of **<3, 'Somvir', 22, 10>** from EMPLOYEE relation is allowed because deletion of this tuple causes no violation.



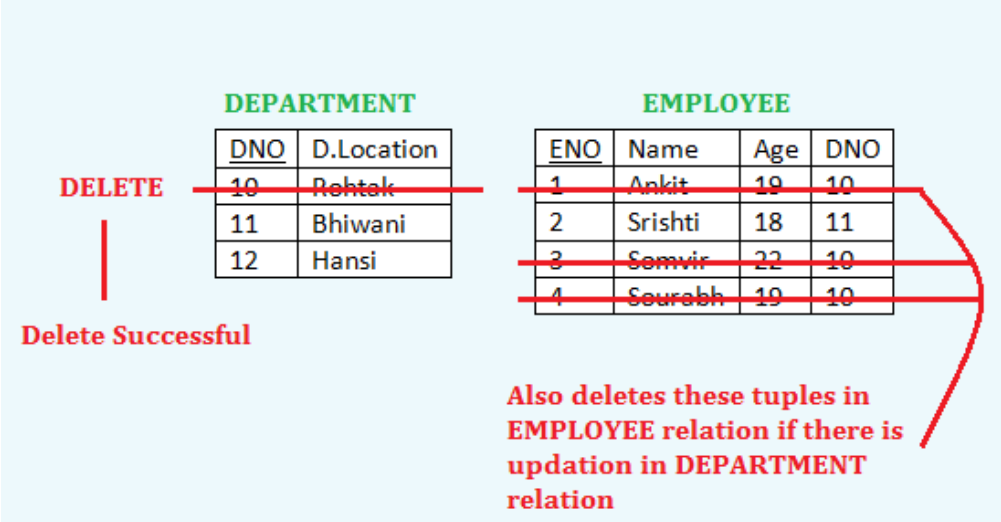
Deletion in a Referenced Relation (DEPARTMENT relation) - There are **three options** available if a deletion causes violation –

Reject the Deletion - (ON DELETE NO ACTION) - It prevents deleting a parent when there are children. It is the the **Default Constraint**. **For example**, Deletion of **<10, 'Rohtak'>** from DEPARTMENT relation is not allowed because deletion of this tuple will violate referential integrity as in the EMPLOYEE relation, the tuple with **DNO=10** causes problem. And therefore, reject the deletion.



Cascade Deletion - (ON DELETE CASCADE) - If deletion causes integrity violation, then delete from both the table i.e. if the tuples are deleted from the referenced table, then the tuple will also be deleted from the referencing relation that is being deleted. **For**

example, Deletion of **<10, 'Rohtak'>** from DEPARTMENT relation will delete the following tuples in EMPLOYEE relation :
<1, 'Ankit', 19, 10>
<3, 'Somvir', 22, 10>
<4, 'Sourabh', 19, 10>



Modify the referencing Attributes - (ON DELETE SET NULL) - sets null value or some valid value in the foreign key field for corresponding deleting referenced value. i.e. changing the referencing attribute values that cause the violation either null or another valid value. If there is no restriction or constraint applied for putting the NULL value in the referencing relation – then allow to delete from referenced relation otherwise prohibited. **For example**, Deletion of **<10, 'Rohtak'>** from DEPARTMENT relation will delete the following tuples in EMPLOYEE relation :
<1, 'Ankit', 19, 10>
<3, 'Somvir', 22, 10>
<4, 'Sourabh', 19, 10>
and null values will be filled in the EMPLOYEE relation on that place. So the relations will look like this :

ON DELETE SETS NULL

DEPARTMENT

DNO	D.Location
~~10~~	~~Rohtak~~
11	Bhiwani
12	Hansi

DELETE

Delete Successful

EMPLOYEE

ENO	Name	Age	DNO
~~1~~	~~Ankit~~	~~19~~	~~10~~
2	Srishti	18	11
~~3~~	~~Somvir~~	~~22~~	~~10~~
~~4~~	~~Sourabh~~	~~19~~	~~10~~

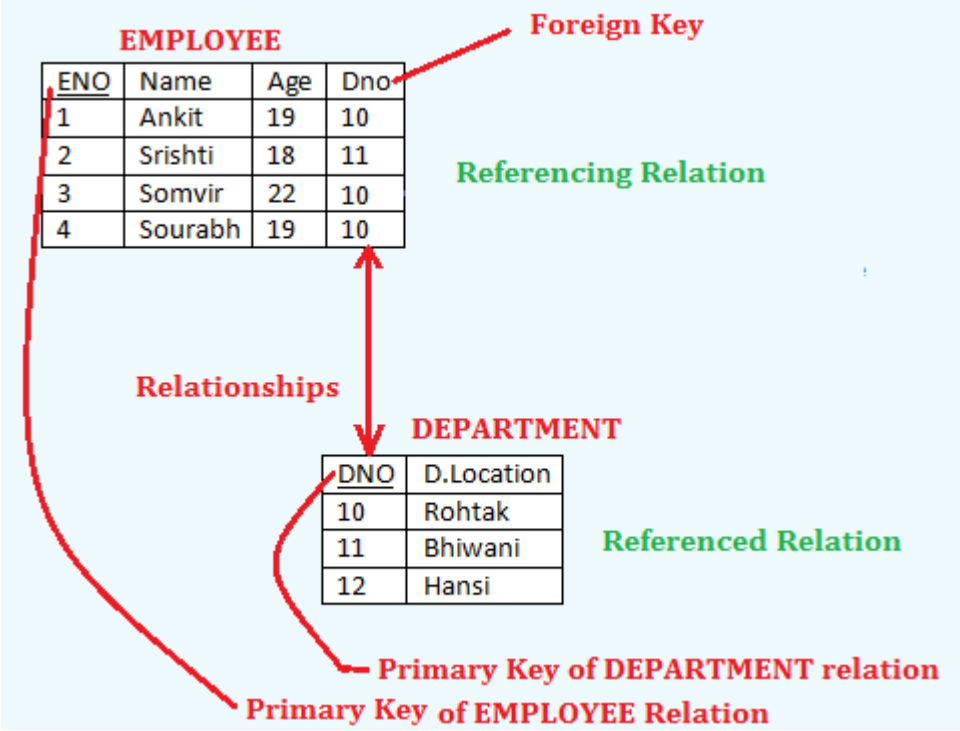
ENO	Name	Age	DNO
NULL	NULL	NULL	NULL
2	Srishti	18	11
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL

Puts NULL in referencing relation if there is deletion in referenced relation

Update Operations and Dealing with Constraint Violations -

The Update Operation :

Consider two existing relations named **EMPLOYEE** and **DEPARTMENT**.

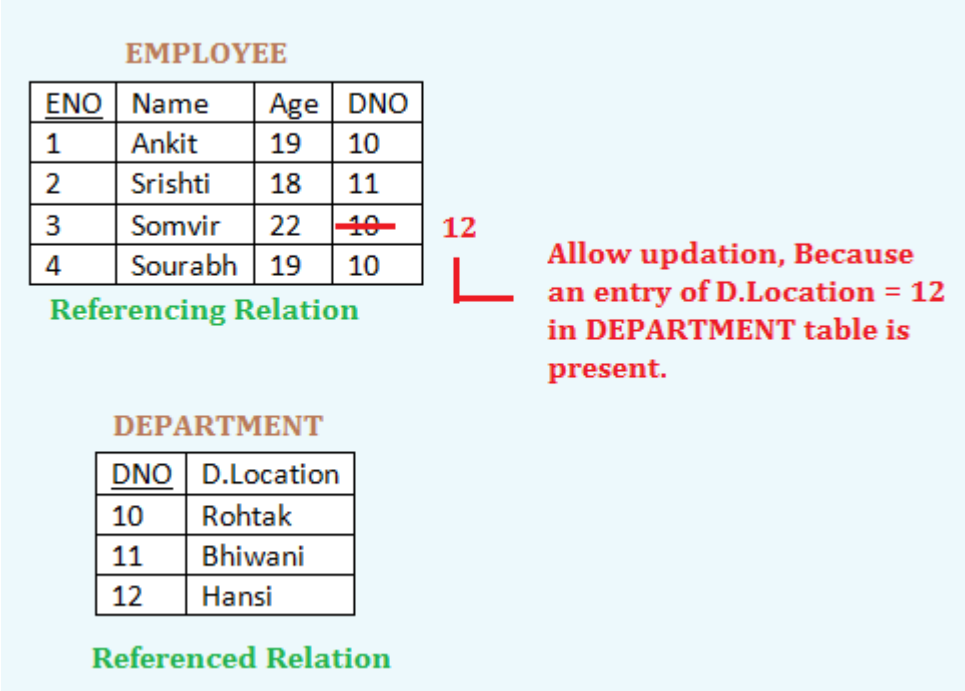


Some Basic Points about the Figure -

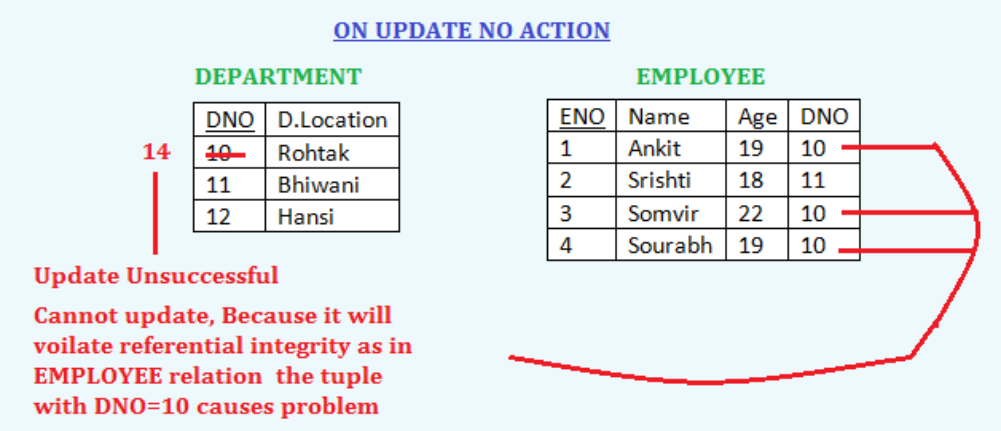
- Here, ENO is a Primary Key and DNO is a Foreign Key in EMPLOYEE relation.
- Table that contain candidate key is called **referenced relation** and
- The table containing foreign key is called **referencing relation**.
- So, the relation **DEPARTMENT is a referenced relation** and
- The relation **EMPLOYEE is a referencing relation**.

The update operation violates only referential Integrity Constraints or Integrity Rule 2.

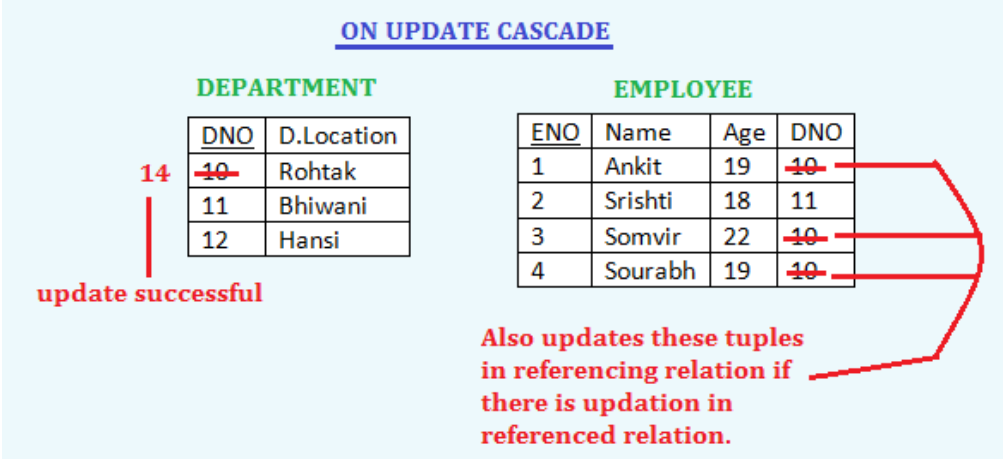
1. **Updation in a referencing relation -** updation of referencing attribute may causes referential integrity violation. The Updation restricted if causes violation. For Example, If there is no violation, then updation will be allowed.



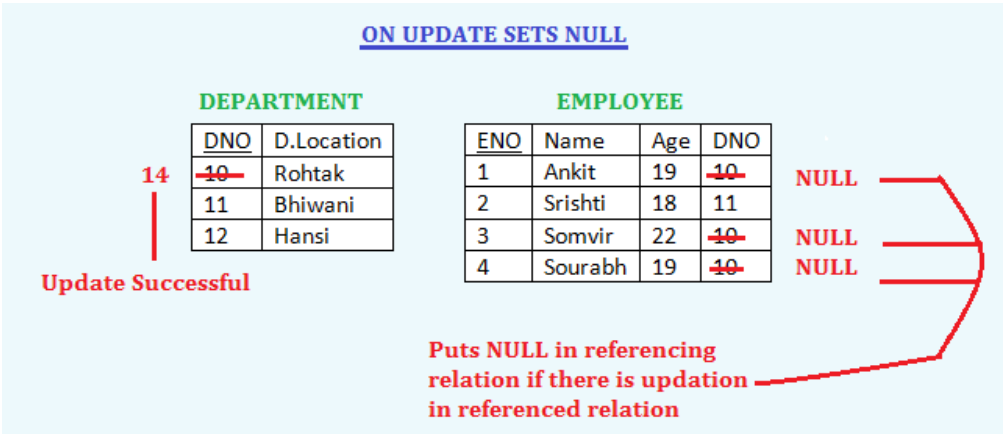
2. **Updation in a referenced relation -** There are again **three options** available if an updation causes violation -
    1. **Reject the updation - (ON UPDATE NO ACTION) -** It prevents updating a parent when there are children. It is the the **Default Constraint**. For example,



2. **Cascade Updation - (ON UPDATE CASCADE) -** If updation causes integrity violation, then update in both the table i.e. if the tuples are updated from the referenced table, then the tuple will also be updates from the referencing relation that is being updated.



3. **Modify the referencing Attributes - (ON UPDATE SET NULL) -** sets null value or some valid value in the foreign key field for corresponding updating referenced value. i.e. changing/updating the referencing attribute values that cause the violation either null or another valid value. If there is no restriction or constraint applied for putting the NULL value in the referencing relation – then allow to update from referenced relation otherwise prohibited.



Questions on Foreign Keys and Constraint Violations -

Question 1 :

Assume one is primary key and other one is Foreign Key. Identify which of the set is the primary key and which of the set is foreign key ?

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

Solution :

Here A is the PRIMARY KEY, and C is the FOREIGN KEY. Because A follows all the properties of primary key. That is,

1) A does not contain any null values.

2) All the values in A are distinct. So A is Primary key.

C is the Foreign key referencing A as it contains the values referencing with A.

Question 2 :

What are the tuples additionally deleted to preserve referential integrity when the tuple(2,4) is deleted ?

Solution :

A	C
~~2~~	~~4~~
3	4
4	3
~~5~~	~~2~~
~~7~~	~~2~~
~~9~~	~~5~~
6	4

Question 3:

Let R(a, b, c) and S(d, e, f) be two relations in which d is the foreign key of S that refers to the primary key of R. Consider the following four operations on R and S

(a) Insert into R

(b) Insert into S

(c) Delete from R

(d) Delete from S

Which of the following statements is true about the referential integrity constraint above?

(A) None of (a), (b), (c), or (d) can cause its violation
(B) All of (a), (b), (c), and (d) can cause its violation
(C) Both (a) and (d) can cause its violation
(D) Both (b) and (c) can cause its violation

Solution :

(D) Both (b) and (c) can cause its violation

Let's take an example,

R		
a	b	c
S1	--	--
S2	--	--
S3	--	--

S		
d	e	f
S1	--	--
S2	--	--

Here 'd' is the foreign key of S and let 'a' is the primary key of R.

a) Insertion into R - will cause no violation as it does not result inconsistency.

b) Insertion into S - may cause violation because there may not be entry of the tuple in relation R. Example entry of <S4,-,-> is not allowed because there is no S4 tuple exists in relation R and it results in inconsistency.

c) Delete form R - may cause violation. For example, Deletion of tuple <S2,-,-> will cause violation as there is entry of S2 in the foreign key table and so produce inconsistency.

d) delete from S - will cause no violation as it does not result inconsistency.