Relational Data Model Properties & Constraints

Dr. Jeevani Goonetillake

UCSC

Relational Model Concepts

- The relational model represents the database as a collection of relations.
- Each relation resembles a table of values.
- Each row in the table represents a collection of related data values.
- All values in a column are of the same data type.

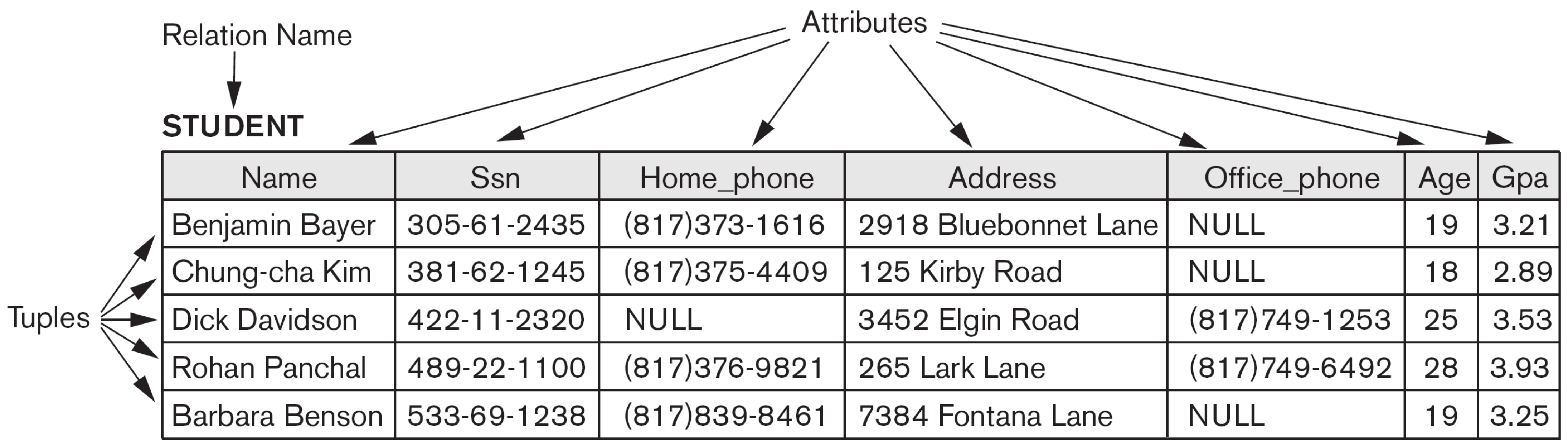
Terminology

- a row is called a tuple (Row → Tuple)
- a column header is called an attribute (Column → Attribute)
- the table is called a relation (Table → Relation)
- the data type describing the types of values that can appear in each column is represented by a domain (Data Type → Domain)

Relational Model Concepts

Concepts

Table, Row, Column, Data Type : Relation, Tuple, Attribute, Domain



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Relational Model Concepts

- A relation schema $R(A_1, A_2, \ldots, A_n)$ is made up of a relation name R and a list of attributes $A_1, A_2, \ldots, A_n$.
- The degree (or arity) of a relation is the number of attributes n of its relation schema.
- A relation schema is used to describe a relation; R is called the name of this relation.
- Each attribute A_i is the name of a role played by some domain D in R
- D is called the domain of A_i and denoted by $dom(A_i)$.

Relational Model Concepts

- A domain D is a set of atomic values.
- Atomic: each value in the domain is indivisible as far as the relational model is concerned.
- Specified by use of a data type from which the data values forming the domain are drawn.
- Domain is given a name, data type, and format

  e.g. Salary DECIMAL (9,2)

  This defines the salary column with 9 digits which include 2 digits after the decimal point: 9999999.99.

  GPA FLOAT (between 0 and 4)

  Dept_Name {Computer Science, Economics, Physics}

Properties Of Relational Model Concepts

- A relation is defined as a *set* of tuples – hence should not have duplicate **tuples.**
- Each tuple is unique.
  - No two tuples in a relation are identical.
- The sequence of columns (left to right) is insignificant.
  -The *attribute name* appears with its *value* as (<attribute>, <value>) pairs with respect to a tuple. The columns of a relation can be interchanged within a tuple without changing the meaning or use of the relation.

Properties Of Relational Model Concepts

- A relation is defined as a *set* of tuples – hence should not have duplicate **tuples.**
- Each tuple is unique.

  - No two tuples in a relation are identical.
- The sequence of attributes/columns (left to right) is insignificant.

  -The *attribute name* appears with its *value* as (<attribute>, <value>) pairs with respect to a tuple. The columns of a relation can be interchanged within a tuple without changing the meaning or use of the relation.

Properties Of Relational Model Concepts

- Tuple ordering is not part of a relation definition because a relation attempts to represent facts at a logical or abstract level. Many tuple orders can be specified on the same relation.
- In other words, a relation is not sensitive to the ordering of tuples.
- However, when a relation is implemented as a table, a file is created.
- In a file, records are physically stored on disk (or in memory), so there always is an order among the records. This ordering indicates first, second, i^{th}, and last records in the file.

Entities & Relationships as Relations

- Notice that some relations may represent facts about entities, whereas other relations may represent facts about relationships.
- For example, a relation Works_On (<u>Empid, Projid</u>, No_hrs) provides that employees are working certain number of hours on a particular project.
- A tuple in this relation relates an employee to his/her project.
- Hence, the relational model represents facts about both entities and relationships uniformly as relations.

Entities & Relationships as Relations

- This sometimes compromises understandability because one has to guess whether a relation represents an entity type or a relationship type.
- The mapping procedures of entity–relationship (ER) model show how different constructs of the ER/EER conceptual data models get converted to relations.

Relation Vs Table

- A table is a representation of a relation but the two are not strictly equivalent.
- A relation contains sets (no duplicates allowed) while a table contains bags (duplicates allowed).
- Uniqueness with respect to rows of a table is enforced through unique key constraint.
- Relation and table would be equivalent if the ordering of rows is not significant, and the table has no duplicate rows.

Properties of Table

- Each table (or relation) in a database has a unique name.
- Each column (or attribute) within a table has a unique name.
- An entry at the intersection of each row and column is atomic (or single-valued).
- There can be no multi-valued attributes in a relation and thus in turn in a table. If there are multi valued attributes they have to be handled in accordance with the limitations present in the relational model.

Employee

EmpNo	Ename	Designation
179	Silva	Manager
857	Perera	Programmer
342	Dias	Clerk

Relational Model Constraints

- Inherent model-based constraints (or implicit constraints)

  Refers to the constraints associated with model itself.

- Schema-based constraints (or explicit constraints)

  Constraints that can be directly expressed in the schemas of the data model, typically by specifying them in the DDL.

- Application-based constraints (or business rules)

  Enforced on DB using application program or rules/triggers.

  Examples : The salary of an employee should not exceed the salary of the employee's supervisor.

Relational Model Constraints

- Inherent model-based constraints (or implicit constraints)

  Refers to the constraints associated with model itself.

- Schema-based constraints (or explicit constraints)

  Constraints that can be directly expressed in the schemas of the data model, typically by specifying them in the DDL.

- Application-based constraints (or business rules)

  Enforced on DB using application program or rules/triggers.

  Examples :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 40.

Inherent Model-based Constraints

These are assumed to hold by the definition of that model (built into the system and not specified by a user).

- Inherent constraints
  - A relation consists of a certain number of simple attributes.
  - An attribute value is atomic
  - No duplicate tuples are allowed

Inherent Model-based Constraints

- Each value in a tuple is an atomic value; that is, attribute value is not divisible into components within the relational model.
- Hence, composite and multivalued attributes cannot be represented.
- This model is sometimes called the flat relational model.
- The theory behind the relational model was developed based on the first normal form assumption.
- As a result, multivalued attributes must be represented by separate relations, and composite attributes are represented only by their simple component attributes in the basic relational model.

Schema-based constraints

- The schema-based constraints include
  - domain constraints,
  - key constraints,
  - constraints on NULLs,
  - entity integrity constraints and
  - referential integrity constraints.

Domain Constraints

- Specifies that the value of each attribute ‘A’ must be an atomic value and from the specified domain dom(A).
- The data types associated with domains typically include standard numeric data types for integers (such as short integer, integer, and long integer) and real numbers (float and double-precision float).
- Characters, Booleans, fixed-length strings, and variable-length strings are also available, as are date, time, timestamp, and other special data types.

Domain Constraints

- Domains can also be described by a subrange of values from a data type or as an enumerated data type in which all possible values are explicitly listed.

```
Salary Decimal(9,2) Check (Salary >= 100000),

CREATE TABLE Person (
   NIC char(10) NOT NULL,
   LastName varchar(150) NOT NULL,
   FirstName varchar(150),
   Age int,
   City varchar(100),
   CONSTRAINT CHK_Person CHECK (Age>=18 AND
                                              City='Colombo')
);
```

```
CREATE TABLE Ticket (
Tid INT PRIMARY KEY AUTO_INCREMENT,
Title varchar(255) NOT NULL,
Priority ENUM('Low', 'Medium', 'High') NOT NULL
);
```

```
CONSTRAINT priority_col CHECK (Priority IN
                              ('Low', 'Medium', 'High'))
```

Domain Constraints

- **UNIQUE** constraint ensures that all values in a column are different.

- Both UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

```
Dname VARCHAR(15) UNIQUE,
```

Key Constraints

- No two tuples should have the same combination of values for their attributes.
- The value of a key attribute can be used to identify uniquely each tuple uniquely in the relation.
- This property is **time-invariant.**
- If a relation has more than one key, they are called candidate keys. One of them is chosen as the primary key.

Entity Integrity Constraints

- The **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 it is not possible to identify tuples uniquely in the database.

Constraints on NULLs

- Another constraint on attributes specifies whether NULL values are permitted for that attribute.
- For example, if every Employee tuple must have a valid, non-NULL value for the Name attribute, then Name of Employee is constrained to be NOT NULL.

Relational Constraints

```
CREATE TABLE Employee (
 Emp_id CHAR(05) PRIMARY KEY,
 Emp_name VARCHAR(55) NOT NULL,
 Hire_date DATE NOT NULL,
 NID CHAR(10) NOT NULL UNIQUE,
 Salary DECIMAL (9,2)  NOT NULL
          CHECK (Salary >= 30000 AND Salary <= 100000)
 );
```

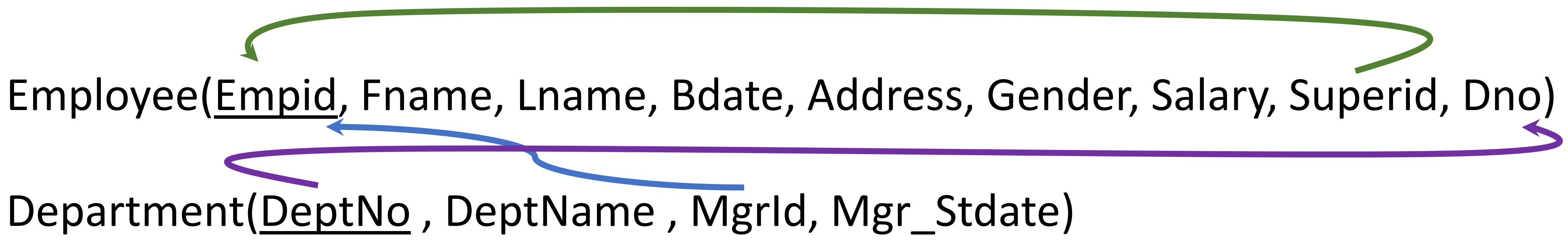
```
CREATE TABLE WorksFor(
Emp_id CHAR(05),
Proj_id CHAR(03),
No_hours INT,
PRIMARY KEY (Emp_id, Proj_id));
```

Relational Constraints

```
CREATE TABLE Employee (
    Empno CHAR(6) NOT NULL Constraint Emp_pk PRIMARY KEY,
    Firstname VARCHAR(200) NOT NULL,
    Lastname VARCHAR(200) NOT NULL,
    Salary DECIMAL(9,2) Constraint Sal_ck CHECK (Salary >= 100000),
    Bonus DECIMAL(9,2),
    Tax DECIMAL(9,2),
            Constraint Bonus_ck CHECK (Bonus > Tax)
    );
```

Referential Integrity Constraints

- The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations.
- Informally what this means is that a tuple in one relation that refers to another relation must refer to an existing tuple.
- To define referential integrity the concept of foreign keys is used.



Employee(Empid, Fname, Lname, Bdate, Address, Gender, Salary, Superid, Dno)

Department(DeptNo , DeptName , MgrId, Mgr_Stdate)

Foreign Key

- An attribute in a relation of a database that serves as the primary key of another relation in the same database.

  Employee(Emp_No, Emp_Name, **Dno**)

  Department(**Dept_No**, Dept_Name, Mgr_No)

=== works for ==>

Referential Integrity Constraints

Employee (First Table)

Foreign Key

Emp_ID	**Name**	**Salary**	**Designation**	**Dep_ID**
1234	Isuru	50,000	Clerk	70
2345	Kapila	70,000	Manager	20
3456	Gihan	90,000	Director	50
4567	Supun	160,000	Programmer	

This value is not allowed because this value is not defined as a PK in the Department table

The value can be **NULL** as the Employee(Supun) may not have any Department.

Primary Key

This value cannot be **NULL** as you will not be able to identify employees uniquely

Primary Key

Department (Second Table)

Dept_ID	**DeptName**
70	Finance
50	Marketing

Referential Integrity Constraints

```
CREATE TABLE Employee (
  emp_id CHAR(04) NOT NULL PRIMARY KEY,
  emp_name VARCHAR(55) NOT NULL,
  hire_date DATE NOT NULL,
  salary INT NOT NULL
            CHECK (salary >= 30000 AND salary <= 100000),
  dept_id INT,
  FOREIGN KEY (dept_id) REFERENCES department(dept_id)
);
```

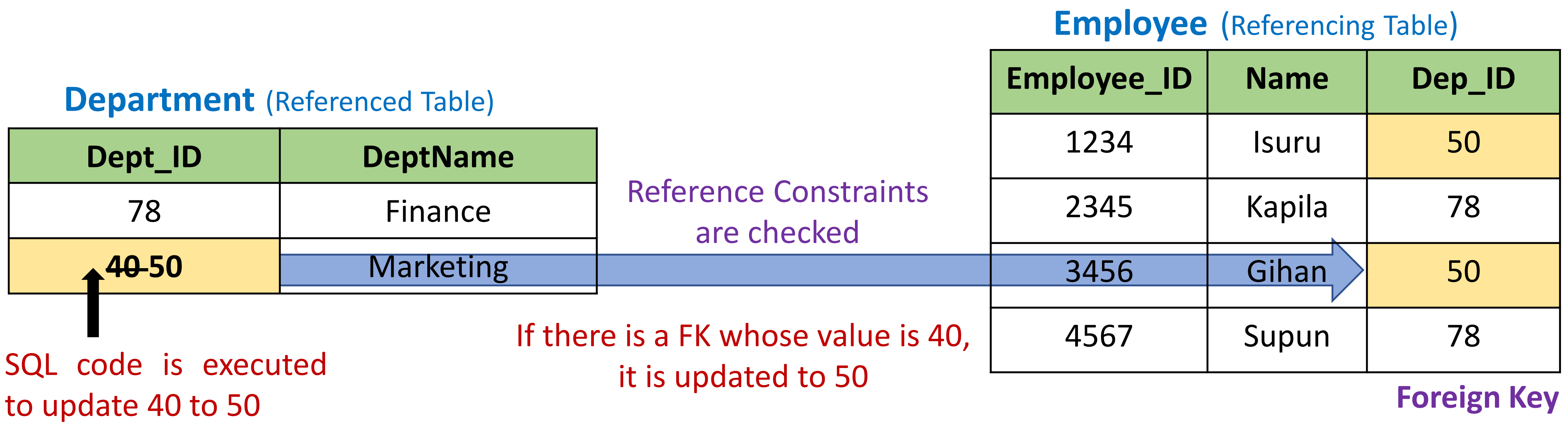
Referential Constraint Actions

- Referential constraint may contain a referential action which is an insertion, update, or deletion action in the referenced table, and it is specified with CASCADE, SET DEFAUT, SET NULL or RESTRICT.

Referential Constraint Actions

When **UPDATE CASCADE** is specified

- A foreign key with **UPDATE CASCADE** means that if primary key of the parent/referenced table is changed, the corresponding foreign key in the child/referencing table is also changed.



Department (Referenced Table)

Dept_ID	DeptName
78	Finance
~~40~~ 50	Marketing

SQL code is executed to update 40 to 50

Reference Constraints are checked

If there is a FK whose value is 40, it is updated to 50

Employee (Referencing Table)

Employee_ID	Name	Dep_ID
1234	Isuru	50
2345	Kapila	78
3456	Gihan	50
4567	Supun	78

Foreign Key

Referential Constraint Actions

When **UPDATE CASCADE** is specified

- In **Employee** Table

```
CONSTRAINT Emp_Dep_FK
FOREIGN KEY (Dep_ID) REFERENCES Department (Dept_ID)
ON UPDATE CASCADE

UPDATE Department SET Department_ID = 40
WHERE Department_ID = 50
```

Updating a department ID will result in changing it in the Employee table (Update with new department ID for the employee working for them).

Referential Constraint Actions

When **DELETE RESTRICT** is specified

- A foreign key with **DELETE RESTRICT** means that a tuple of the referenced table is tried to be deleted, that will fail if the primary key of that tuple is referred by some tuples in another table.

Department (Referenced Table)

Dept_ID	DeptName
78	Finance
50	Marketing

SQL code is executed to delete 50

Reference Constraints are checked

If there is a FK whose value is 50, a constraint violation results (The PK is not deleted)

Employee (Referencing Table)

Employee_ID	Name	Dep_ID
1234	Isuru	50
2345	Kapila	78
3456	Gihan	50
4567	Supun	78

Foreign Key

Referential Constraint Actions

When **DELETE RESTRICT** is specified

- In **Employee** Table

```
CONSTRAINT Emp_Dep_FK
FOREIGN KEY (Dep_ID) REFERENCES Department (Dept_ID)
ON DELETE RESTRICT

DELETE FROM Department
WHERE Department_ID = 50
```

A department tuple can only be deleted if a tuple relevant to that is not found in employee table (i.e. if there are no employees working for it).

Modification Operations

- Insert
- Delete
- Update

Insert Operation

Insert can violate the following constraints.

- Entity integrity constraint (null for the PK)
- Key constraint (PK already exists)
- Referential Integrity constraint (referring a tuple that does not exist)
- Domain Constraint (value does not appear in the
  corresponding domain)

Insert Operation

Examples

- insert <null, 'Thilini', 60000,'Secretory', 70 >

  -> violates the entity integrity constraint!

- insert <2345, 'Supun', 80000,'Manager', 50 > -> violates the key constraint!

insert <4567, 'Pasindu', 90000,'Sales Rep', 90 >

  -> violates the referential integrity constraint!

- insert <5678, 'Isuru', 35000,'Staff Assistant', 70>

  -> violates the domain constraint!

Salary ≥ 40000 (salary values should be equal or above Rs. 40000)

Employee (Referencing Table)

Emp_ID	Name	Salary	Desination	Dep_ID
1234	Prasad	45,190	Clerk	70
2345	Kapila	63,280	Manager	70
3456	Kasun	69700	Director	50

Department (Referenced Table)

Dept_ID	Dep_Name
70	Finance
50	Marketing

Delete Operation

- Can violate only referential integrity.
  - The tuple/row involved is referenced by tuples from other the relations/tables.

Examples

- Delete the Department tuple with Dept_ID = 70

- Available options
  - Modify the referencing attribute values to a default value or NULL

    (ON DELETE SET DEFAULT, ON DELETE SET NULL)
  - Reject violations ***(ON DELETE RESTRICT)***
  - Cascade (propagate)

    (ON DELETE CASCADE)

Employee

Emp_ID	Name	Salary	Designation	Dep_ID
1234	Prasad	45,190	Clerk	70
2345	Kapila	63,280	Manager	70
3456	Kasun	69700	Director	50

Department

Dept_ID	Dep_Name
70	Finance
50	Marketing

Update Operation

- Updating an attribute that is neither a primary key nor a foreign key usually causes no problems.
- Then the only requirement is to check whether the new value is of the correct data type and domain. That is to check any violation of domain constraints.

  - Update the salary of the Employee tuple with Emp_ID = ‘1234’ to Rs.35,000 when all salary values are required be ≥ 40,000.

  -> violates the domain constraint!

```
Salary DECIMAL(9,2) Constraint Sal_ck CHECK (Salary >= 40000)
```

Update Operation

- **Modifying** a primary key value is similar to **deleting** one tuple and **inserting** another in its place.
- If a foreign key attribute is modified, the DBMS must make sure that the new value refers to an existing tuple in the referenced relation (or could be null if null is permitted).

Examples

- update the Dep_ID to 60 of the Employee tuple with Emp_ID = ‘2345’
  -> violates the referential integrity constraint!
- update the Emp_ID to ‘3456’ of the Employee tuple with Emp_ID = ‘1234’
  -> violates the key constraint!