The Relational Data Model

Dr. Enosha Hettiarachchi

Structure

- Introduction to Relational Data Model:
  - Review of database models
  - Definition of
    - Relation
    - Attribute
    - Tuple
    - Domain
    - Instance
    - Cardinality
    - Degree
    - Schema
    - Constraints

Structure

- Concepts of keys:
  - Candidate key
  - Primary key
  - Alternate key
  - Composite key
  - Surrogate key
  - Foreign key
- Fundamental integrity rules:
  - Entity integrity
  - Referential integrity
  - Domain constraints
  - Key constraints

Introduction to Relational Data Model

Relational Model

Data is presented to the user as tables:

- Tables are comprised of ***rows*** and a fixed number of named ***columns***.

Table

	Column 1	Column 2	Column 3	Column 4
Row				
Row				
Row				

Relational Model

- Columns are attributes describing an entity. **Each column must have a unique name and a data type.**

Employee

Name	Designation	Department

Row

Row

Row

Structure of a relation (e.g. Employee)

Employee(Name, Designation, Department)

Relational Model

- Rows are records that present information about a particular entity occurrence

Employee

	Name	Designation	Department
Row	De Silva	Manager	Personnel
Row	Perera	Secretary	Personnel
Row	Dias	Manager	Sales

Relational Model Terminology

- **Row** is called a '**tuple**'
- **Column header** is called an '**attribute**'
- **Table** is called a '**relation**'
- The **data type describing the type of values** that can appear **in each column** is called a '**domain**'.

Relational Model Terminology

- Eg:-
  - Names : the set of names of persons
  - Employee_ages : value between 15 & 80 years old

  The above is called ‘logical definitions of domains’.

  A data type or format can also be specified for each domain.

  Eg: The employee age is an **integer** between 15 and 80

Properties

- Each relation (or table) in a database has a unique name
- Each attribute (or column) 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

Properties

- Each row is unique;

  no two rows in a relation are identical

- The sequence of columns (left to right) is insignificant;

  the columns of a relation can be interchanged without changing the meaning or use of the relation.

Properties

- The sequence of rows (top to bottom)

  is insignificant;

  rows of a relation may be interchanged

  or stored in any sequence.

Domain

- A *domain* is the original set of atomic values used to model data.
- *atomic value* - each value in the domain is indivisible as far as the relational model is concerned.
- For example:
  - The domain of Marital Status has a set of possibilities: Married, Single, Divorced.
  - The domain of Shift has the set of all possible days: {Mon, Tue, Wed…}.
  - The domain of Salary is the set of all floating-point numbers greater than 0 and less than 200,000.
  - The domain of First Name is the set of character strings that represents names of people.
- In summary, a domain is a set of acceptable values that a column is allowed to contain. This is based on various properties and the data type for the column.

Cardinality of the domain

- The **cardinality of a domain** refers to the number of unique, permissible values that an attribute (or column) can take.
- The domain itself represents the set of all valid values that an attribute can store, while its cardinality indicates the size of this set.
- For example:
  - If a domain is defined as {Red, Blue, Green}, its **cardinality is 3,** as it has three possible values.
  - The cardinality of a domain for the "days of the week" attribute is 7, as there are seven unique values: {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}.
  - If a domain represents all possible integers, its cardinality would be infinite, as there are endless integer values.

Relation schema and Attributes

- **Relation Schema**:
  - A relation schema defines the structure of a specific table (or relation) in a database.
  - It outlines what information the table holds by specifying the table's name and listing its attributes (columns).
  - Think of the relation schema as the blueprint for a table, describing the type of data stored and how it is organized, but not the actual data itself.
- For example, a relation schema for a "Student" table could be defined as:
  - **Student (StudentID, Name, Age, Major)**
  - This schema indicates that the "Student" table will have columns for StudentID, Name, Age, and Major.

Relation schema and Attributes

- **Attributes**:
  - Attributes are the individual columns listed in a relation schema, each representing a specific type of information that the table stores.
  - Each attribute has a domain, which defines the set of possible values it can hold (like integers, text, or dates).
  - In the example of the "Student" table:
    - *StudentID could be an integer representing a unique identifier for each student.*
    - *Name might be a string storing each student's name.*
    - *Age could be an integer for the student's age.*
    - *Major could be a string that specifies the student's field of study.*

Together, the relation schema and its attributes create a structured way to store, organize, and access data within a relational database.

Degree of a relation

- The **degree** of a relation (or table) refers to the number of attributes (columns) it has.
- For example:
  - If a table “Employee” has attributes EmployeeID, Name, Position, and Salary, the **degree of this relation is 4**, because it has four attributes.
  - A simple table “Product”, with only two attributes, like ProductID and Price, has a **degree of 2**.

Degree of STUDENT relation is 7.

Relation Instance

- An **instance of a relational table** refers to the actual data contained in that table at a specific point in time. It is a collection of rows (also called tuples) that follow the structure defined by the table's schema.

- The relational schema defines the attributes (columns) and their data types, **while an instance of the table is the set of records (data values) for these attributes.**

- For example, if we have a schema for a table called Employee defined as:
  - Employee(EmployeeID, Name, Position, Salary)

Relation Instance

- An instance of this table might look like the following:

EmployeeID	Name	Position	Salary
1001	Alice	Manager	60000
1002	Bob	Developer	50000
1003	Carol	Analyst	55000

- In this example:
  - The schema is **Employee(EmployeeID, Name, Position, Salary).**
  - The instance is the actual data shown in the table at this point in time (three rows of data).

- The instance can change over time as data is added, updated, or deleted, but the schema remains constant, defining the structure.

Cardinality of a Table

- The **cardinality of a table** refers to the number of rows (or tuples) in that table. It is essentially a measure of how many individual records are present in the table at any given moment.

- For example:
  - If a table Students has 50 records (rows), the cardinality of the Students table is 50.
  - If a table Orders contains 2000 rows, the cardinality of the Orders table is 2000.

Concepts of keys

Candidate Key

- A **candidate key** is a set of one or more columns (attributes) in a table that can uniquely identify a record (row) within the table. A table can have multiple candidate keys, and each candidate key could potentially serve as the primary key.

- **Purpose**: Candidate keys provide alternatives for uniquely identifying rows in a table. One of them will be selected as the primary key.

- **Example:** In a Student table, both **StudentID and Email can serve as candidate keys because both can uniquely identify each student.**

Candidate Key

- **Properties**:
  - Unique: Each candidate key uniquely identifies a row in the table.
  - Minimal: Each candidate key contains only the necessary attributes to ensure uniqueness.

Primary Key

- A **primary key** is a specific candidate key chosen by the database designer to uniquely identify each record in a table.
- The primary key must contain unique values for each row and cannot have NULL values.

- **Purpose**: Ensures that each record can be uniquely identified and helps optimize data retrieval.

- **Example:** In the Student table**, StudentID could be the primary key because it uniquely identifies each student.**

Primary Key

- **Properties:**
  - Uniqueness: No two rows can have the same value for the primary key.
  - Non-null: A primary key must have a value for every record (cannot be NULL).

Alternate Key

- An **alternate key** is any candidate key that is not chosen as the primary key. These are keys that also have the potential to uniquely identify records but are not selected for use as the primary key.

- **Purpose**: Provides alternative ways to uniquely identify rows in the table.

- **Example:** In the Student table, if StudentID is chosen as the primary key, then Email could be an alternate key because it also uniquely identifies students.

Alternate Key

- **Properties:**
  - Uniqueness: Like the primary key, alternate keys ensure that values are unique.
  - Can allow NULL values: Unlike the primary key, alternate keys may allow NULL values (but typically only one NULL value is allowed).

Composite Key

- A **composite key** is a combination of two or more columns (attributes) that together uniquely identify a row in a table. None of the individual columns in the composite key can uniquely identify a record on their own.

- **Purpose**: Used when a single column is insufficient to uniquely identify records in the table.

- **Example:** In an Order table, a composite key might consist of OrderID and ProductID because no single column uniquely identifies an order. Together, they form a unique identifier for each record.

Composite Key

- **Properties**:
  - Composed of multiple columns.
  - Ensures uniqueness when combined but not individually.

Surrogate Key

- A **surrogate key** is a system-generated, artificial key used to uniquely identify a record in a table. It has no business meaning and is typically created automatically by the database system.

- **Purpose**: Used when natural (business) keys are not suitable or when there is no obvious unique identifier.

- **Example:** An EmployeeID that is auto-incremented for each new employee record is a surrogate key. It has no real-world meaning but is used to uniquely identify each employee.

Surrogate Key

- **Properties**:
  - Typically numeric and auto-incremented.
  - Does not have any inherent meaning or relation to the real-world data.

Foreign Key

- A **foreign key** is a column (or a set of columns) in one table that links to the primary key (or unique key) in another table. It establishes a relationship between two tables by ensuring that the value in the foreign key column exists in the referenced primary key column of the other table.

- **Purpose**: Enforces referential integrity by ensuring that relationships between tables are maintained and that records in one table correspond to valid records in another table.

- **Example:** In an Order table, the CustomerID could be a foreign key that references the CustomerID in the Customer table. This ensures that each order is associated with a valid customer.

Foreign Key

- **Properties:**
  - Ensures referential integrity: Values in the foreign key must match values in the referenced primary or unique key.
  - May allow NULL values, depending on the relationship.

Summary of Key Types:

- **Candidate Key**: A set of columns that can uniquely identify a row. There may be multiple candidate keys in a table.
- **Primary Key:** A candidate key chosen to uniquely identify each row in the table. It cannot have NULL values.
- **Alternate Key**: A candidate key that is not selected as the primary key.
- **Composite Key**: A key made up of multiple columns, used when a single column cannot uniquely identify records.
- **Surrogate Key**: A system-generated key with no real-world meaning, used for uniquely identifying records.
- **Foreign Key**: A key that creates a link between two tables by referencing the primary key of another table.

Fundamental integrity rules

Relational Model Constraints

- **Inherent model based constraints**

  Refers to the constraints associated with model itself.

- **Schema-based constraints**

  Constraints that can be specified on the schema using DDL/SQL.

- **Application-based constraints**

  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 constraints (Schema Based)

- **Domain constraints**
  - specifies that the value of each attribute ‘A’ must be an atomic value (or single-valued) and from the specified domain.
- **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 in the relation and this property is *time-invariant*.

Relational constraints (Schema Based)

- If a relation has more than one key, they are called *candidate keys*.
- One of them is chosen as the *primary key*.

Relational constraints (Schema Based)

Entity Integrity

- ***Primary Key:*** An attribute (or combination of attributes) that uniquely identifies each row in a relation.

  Employee(Emp_No, Emp_Name, Department)

- ***Composite Key:*** A primary key that consists of more than one attribute.

  Salary(Emp_No, Eff_Date, Amount)

◆**Entity integrity constraint specifies that no primary key can be null.**

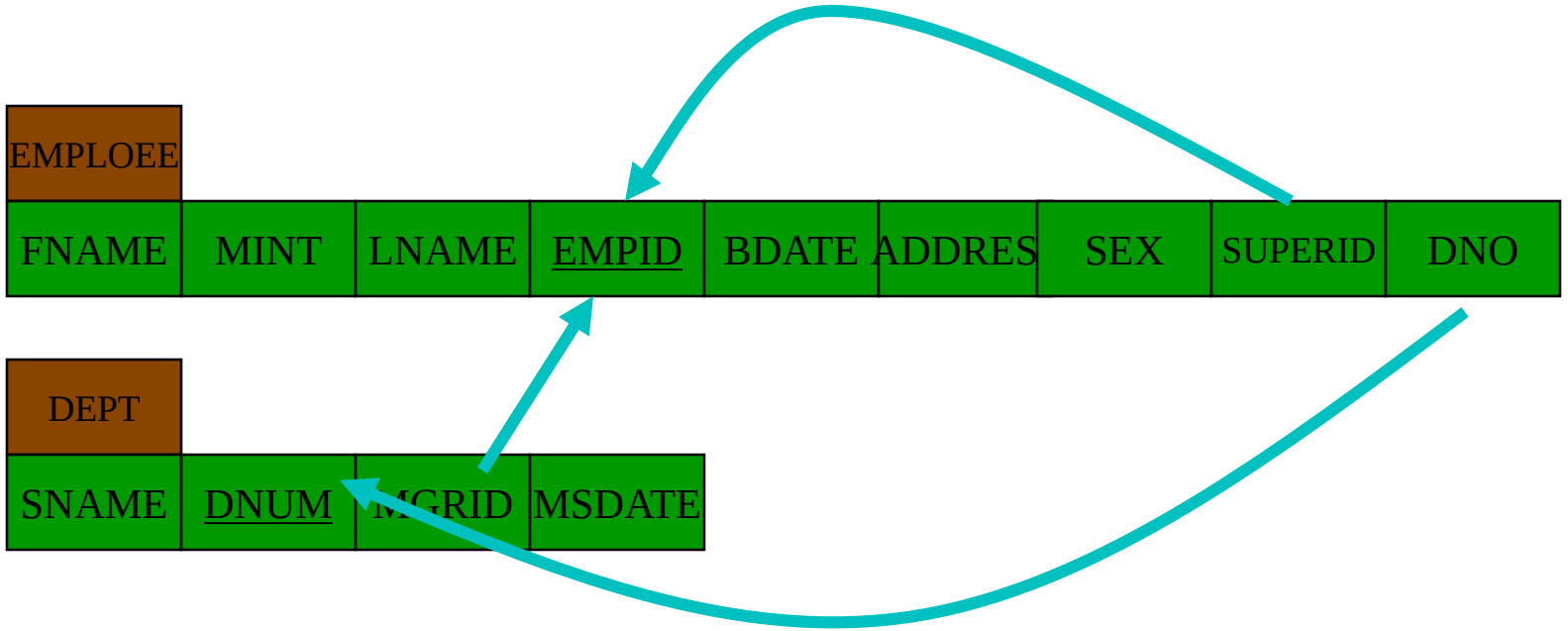
Relational constraints (Schema Based)

- **Referential Integrity**

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations.

Referential integrity/Foreign keys

- 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, we use the concept of foreign keys.



Relational Objects

Relationship

- ***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, *Dept_No)*

Department(Dept_No, Dept_Name, M_No)

=== works for ==>

Relational Objects

- A ***foreign key*** is a set of columns in one table that serve as the primary key in another table.

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

Primary Key

Foreign Key

Department

D-No	D-Name	M-No
4	Finance	857
7	Sales	179

Primary Key

Relational Database State

- Relational database state is the union of all the relation states

COMPANY Database

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

Update Operations on Relations

- Whenever the database is changed/updated, a new database state arises

- Basic operations for updating the database:
  - **INSERT a new tuple into a relation**
  - **DELETE an existing tuple from a relation**
  - **UPDATE/MODIFY an attribute of an existing tuple in a relation**

- Integrity constraints must not be violated by the update operations
  - Updates may propagate to cause other updates automatically. This may be necessary to maintain integrity constraints.

Example

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0

If we change the SSN of John Smith to 111111111, this may require updates to some of the tuples in Works_ON relation (depending on the update option used)

Possible violations for INSERT

- Domain constraint
  - if one of the attribute values provided for the new tuple is not of the specified attribute domain
- Key constraint
  - if the value of a key attribute in the new tuple already exists in another tuple in the relation
- Entity integrity
  - if the primary key value is null in the new tuple
- Referential integrity
  - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation

Example

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

- Insert <‘John’, ‘M’, ‘Doe’, NULL, ‘1977-01-01’, ‘123 Main, TX’, ‘M’, 45000, NULL, 4> into EMPLOYEE will violate entity constraint
- Insert <‘Mary’, ‘M’, ‘Doe’, 123456789, ‘1977-01-01’, ‘123 Main, TX’, ‘M’, 45000, NULL, 4> into EMPLOYEE will violate key constraint
- Insert <‘Tom’, NULL, ‘Doe’, ‘444444444’, 1957, NULL, ‘M’, ’100K’, NULL, 4> into EMPLOYEE will violate domain constraint

Example

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0

```
CREATE TABLE WORKS_ON (
    Essn  INT,
    Pno INT,
    Hours FLOAT,
    PRIMARY KEY (Essn, Pno),
    FOREIGN KEY Essn REFERENCES EMPLOYEE(Ssn)
    FOREIGN KEY Pno REFERENCES PROJECT(Pnum));
```

- Insert <999999999,1,34.5> into WORKS_ON

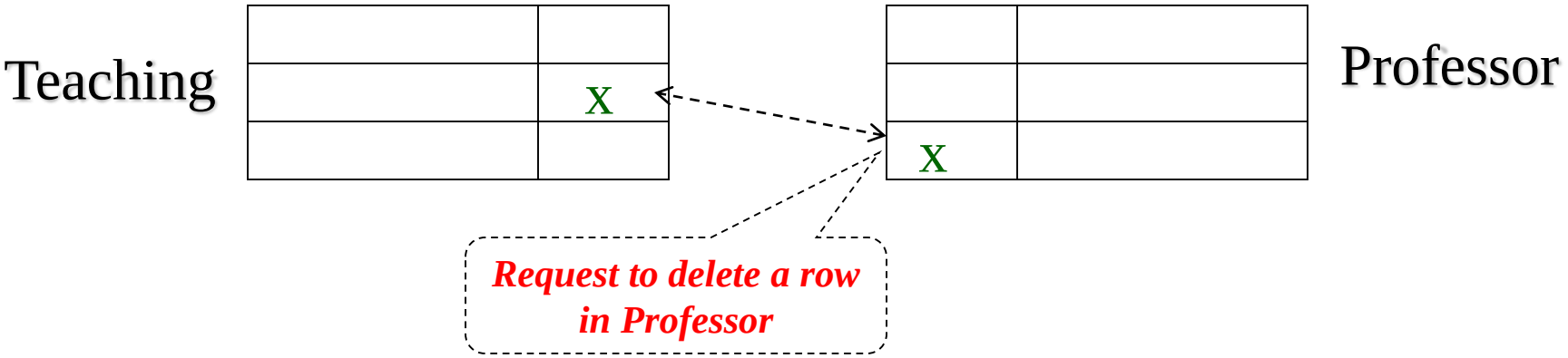
will violate referential integrity constraint

Possible violations for DELETE

- DELETE may violate only referential integrity:
  - If the primary key value of the tuple being deleted is referenced by other tuples in the database
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL
  - One of the above options must be specified for each foreign key constraint

Example

- Deletion from referenced relation Professor:



  - RESTRICT: Reject if row in Teaching references row to be deleted
  - SET NULL: Set value of foreign key in Teaching to NULL
  - SET DEFAULT: Set value of foreign key in Teaching to default value
  - CASCADE: Delete referencing row(s) in Teaching as well

Possible violations for UPDATE

- Domain, key, entity integrity, and referential integrity constraints may be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - Similar to a DELETE operation followed by an INSERT operation
    - Need to specify similar options to DELETE
  - Updating a foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints

END