DBMS: Lecture 1
(Relational Constraints)

By

Dr. Sumit Kumar Tetarave

Domains, Attributes Tuple and Relation

Table/Relation name: PERSON

PERSON_ID	NAME	AGE	ADDRESS	TELEPHONE
1	Sumit	38	Sampark Vihar	12345
2	Amit	35	Sailesh Vihar	67891
3	Rakesh	28	Nandan Vihar	23456

- **Attribute:** The name of each column in a table is used to interpret its meaning and is called an attribute.
- **Tuple:** Each row in a table represents a record and is called a tuple.
  - A table containing "n" attributes in a record is called n-tuple.
- Each table is called a **relation**.
  - For example, the *Figure* represents a relation PERSON.
    - The columns PERSON_ID, NAME, AGE,ADDRESS and TELEPHONE are the attributes of PERSON
    - Each row in the table represents a separate tuple (record).
- **Domain:** A domain is a set of permissible values that can be given to an attribute.
  - So every attribute in a table has a specific domain.
  - Values to these attributes cannot be assigned outside their domains.

Relation

PERSON_ID	NAME	AGE	ADDRESS
1	Sumit	38	Sampark Vihar
2	Amit	35	Sailesh Vihar
3	Rakesh	28	Nandan Vihar

- A relation consists of:
  - Relational Schema
  - Relation instance

- **Relational Schema (**also called *'intension'*)**:** A relational schema specifies the relation's name, its attributes and the domain of each attribute.
  - If R is the name of a relation and A1, A2…An is a list of attributes representing R then R(A1, A2…An) is called a relational schema.
  - Each attribute in this relational schema takes a value from some specific domain called Domain (Ai).
    - For example, the relational schema for relation PERSON will be:

      PERSON(PERSON_ID:integer, NAME: string, AGE:integer, ADDRESS:string)
    - Total number of attributes in a relation denotes the degree of a relation.
    - Since the PERSON relation contains four attributes, so this relation is of **degree 4**.
- **Relation Instance or Relation State (**also called *'extension'*)**:** A relation instance denoted as **r** is a collection of tuples for a given relational schema at a specific point of time.
  - A relation state **r** of the relation schema R ($A_1, A_2, \ldots\ldots A_n$), also denoted by r(R) is a set of n-tuples
    r = {t1,t2,……tm}, where each n-tuple is an ordered list of n values
    t = <v1,v2,……., vn> where each vi belongs to domain (Ai) or contains null values.

  For example: for PERSON table/relation
  - ➢ RELATION INSTANCE- In this instance, m = 3 and n = 4.

Super Keys, Candidate Keys and Primary Keys for the Relations

- A **super key** is an attribute or set of attributes used to identify the records uniquely in a relation.
  - For Example, in the Relation PERSON described earlier PERSON_ID is a super key since PERSON_ID is unique for each person.
  - Similarly (PERSON_ID, AGE) and (PERSON_ID, NAME) are also super keys of the relation PERSON since their combination is also unique for each record.
- **Candidate keys:** Super keys of a relation can contain extra attributes. Candidate keys are minimal super key, i.e. such a key contains no extraneous attribute.
  - An attribute is called extraneous if even after removing it from the key, makes the remaining attributes still has the properties of a key.
  - The following properties must be satisfied by the candidate keys:
    - ✓ A candidate key must be **unique**.
    - ✓ A candidate key's value must **exist.** It cannot be null. (This is also called entity integrity rule)
    - ✓ A candidate key is a minimal set of attributes.
    - ✓ The value of a candidate key must be **stable.** Its value cannot change outside the control of the system.
- A relation can have more than one candidate keys and one of them can be chosen as a **primary key.**
  - For example, in the relation PERSON the two possible candidate keys are
    - PERSON-ID and NAME (assuming unique names in the table).
    - PERSON-ID may be chosen as the primary key.

RELATIONAL CONSTRAINTS

- There are three types of constraints on relational database that include:
  - DOMAIN CONSTRAINT
  - PRIMARY KEY CONSTRAINT
  - INTEGRITY CONSTRAINT
- **Domain Constraint:** It specifies that each attribute in a relation must contain an atomic value only from the corresponding domains.
- The data types associated with commercial RDBMS domains include:
  1) Standard numeric data types for integer (such as short- integer (2B), long integer (4B), long long integer (8B))
  2) Real numbers (float, double precision floats)
  3) Characters
  4) Fixed length strings (eg. CHAR(n)) and variable length strings (eg. VARCHAR).
- **Key Constraint:** This constraint states that the key attribute value in each tuple must be unique, i.e., no two tuples contain the same value for the key attribute.
  - This is because the value of the primary key is used to identify the tuples in the relation.
  - **Example:** In relation PERSON, PERSON_ID is primary key
    - So, PERSON_ID cannot be given as the same for two persons.

Integrity Constraint

Let R be the Table

A#	B	C
Null	B1	C1
A2	B2	C2
Null	B3	C3
A4	B4	C3
A5	B1	C5

- There are two types of integrity constraints:
  - Entity Integrity Constraint
  - Referential Integrity Constraint
- **Entity Integrity Constraint:** It states that **no primary key value can be null**.
  - This is because the primary key is used to identify individual tuple in the relation.
  - So, we will not be able to identify the records uniquely containing null values for the primary key attributes.
  - This constraint is specified on one individual relation.
  - Example: "#" identifies the Primary key of a relation. (Is this valid?)
    - In the relation R above, the primary key has null values in the tuples t1 & t3. NULL value in primary key is not permitted, thus, relation instance is an invalid instance.
- **Referential integrity constraint** It states that the tuple in one relation that refers to another relation must refer to an existing tuple in that relation.
  - This constraint is specified on two relations (not necessarily distinct).

Example: "#" identifies the Primary key of a relation.

"^" identifies the Foreign key of a relation.

R

A#	B	C^
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C3
A5	B1	C5

S

E	C#
E1	C1
E2	C3
E3	C5
E2	C2

- ✓ The value of C^ in every R tuple is matching with the value of C# in some S tuple.
- ✓ If a tuple having values (A6, B2, C4) is added (Are these valid?)
  - ✓ then it is invalid since referenced relation S doesn't include C4. (Violation?)
- ✓ Thus, it will be a violation of **referential integrity constraint**.

Update Operations and Dealing with Constraint Violations

- There are three basic operations to be performed on relations:
  - Insertion
  - Deletion
  - Update
- **The INSERT Operation:** The insert operation allows us to insert a new tuple in a relation.
- When we try to insert a new record, then any of the following four types of constraints can be violated:
  - Domain constraint: If the value given to an attribute lies outside the domain of that attribute.
  - Key constraint: If the value of the key attribute in new *tuple t* is the same as in the existing tuple in *relation R*.
  - Entity Integrity constraint: If the primary key attribute value of new *tuple t* is given as *null*.
  - Referential Integrity constraint: If the value of the foreign key in t refers to a tuple that doesn't appear in the referenced relation.
- **Dealing with constraints violation during insertion:** If the *insertion* violates one or more constraints, then two options are available:
  - Default option: - *Insertion can be rejected and the reason of rejection can also be explained to the user by* DBMS.
  - Ask the user to correct the data, resubmit, also give *the reason for rejecting the insertion*.

Constraint Violations: **Insertion**

PERSON_ID	NAME	AGE	ADDRESS
1	Sumit	38	Sampark Vihar
2	Amit	35	Sailesh Vihar
3	Rakesh	28	Nandan Vihar

- Example: Consider the Relation PERSON

(1) Insert<1, 'Vipin', 20, 'Mayur Vihar'> into PERSON

Violated constraint: - Key constraint Reason: - Primary key 1 already exists in PERSON.

➢ Dealing: - DBMS could ask the user to provide valid PERSON_ID value and accept the insertion if valid PERSON_ID value is provided.

1. Domain constraint
2. Key constraint
3. Entity Integrity Constraint
4. Referential Integrity Constraint

(2) Insert<'null', 'Anurag', 25, 'Patparganj'> into PERSON

Violated constraint: - Entity Integrity constraint Reason: - Primary key is "null".

➢ Dealing: - DBMS could ask the user to provide valid PERSON_ID value and accept the insertion if valid PERSON_ID value is provided.

(3) Insert<'abc', 'Suman', 25, 'IP college'> into PERSON

Violated constraint: - Domain constraint

➢ Reason: - value of PERSON_ID is given a string which is not valid.

**Note:
In all first 3 cases of constraint violations above DBMS could reject the insertion.

(4) Insert <10, 'Anu', 25, 'Patpatganj'> into PERSON Violated constraint: - None

(5) Insert <10, 'Sumit', 25, 'Tetarave'> into PERSON Violated constraint: - But?

Constraint Violations: **Deletion**

- **The Deletion Operation:** Using the delete operation some existing records can be deleted from a relation.
  - To delete some specific records from the database a condition is also specified based on which records can be selected for deletion.
- Only one type of constraint can be violated during deletion,
  - it is referential integrity constraint.
- It can occur when you want to delete a record in the table where it is referenced by the foreign key of another table.

- **Dealing with Constraints Violation**
- If the *deletion* violates referential integrity constraint, then three options are available:
  - Default option: - *Reject the deletion*. It is the job of the DBMS to explain to the user why the deletion was rejected.
  - *Attempt to cascade (or propagate) the deletion* by deleting tuples that reference the tuple that is being deleted.
  - Change the value of *referencing attribute* that causes the violation.

Constraint Violations: **Deletion…**

- **Example:**

R

A#	B	C^
A1	B1	**C1**
A2	B2	C2
A3	B3	C3
A4	B4	C3
A5	B1	C5

Q

E	C#
E1	C1
E2	C3
E3	C5
E2	C2

- Note: 1) "#" identifies the Primary key of a relation. 2) "^" identifies the Foreign key of a relation.

(1) Delete a tuple with C# = "C1" in Q.

Violated constraint: - Referential Integrity constraint Reason: - Tuples in relation R refer to tuple in Q.

Dealing: - Options available are

1) Reject the deletion.
2) DBMS may automatically delete all tuples from relation *Q* and *S* with C # = "C1". This is called cascade detection.
3) The third option would result in putting NULL value in R where C1 exist, which is the first tuple R in the attribute C.

Constraint Violations: **Update**

- **The Update Operations:** Update operations are used for modifying database values.
  - The constraint violations faced by this operation are logically the same as the problem faced by
    - ✓Insertion and
    - ✓Deletion Operations.

Thank You!!!