(For Second Semester)

Relational Data Model : Basic concepts, relational constraints, relational algebra

---

Relational Data Model

In relational data model, a database is represented as a collection of relations. A relation is simply a table of values, where every row represents a collection of related data values.

Basic Concepts of Relational Model

Attribute: Attributes are the properties which define a relation. Each column in a Table is an attribute. For example, Student_Rollno,Student_NAME,etc. are some attributes.

Tables – Each relation represents a table. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

Example-

Student

Roll No.	Name	Address	Class	Section
1	ABC	Ghy	TDC	A
2	BHA	Delhi	HS	B
3	JJGD	Mumbai	TDC	C

Tuple – Each row in a table is called a tuple. A tuple contains a single record.

Student

	Roll_No.	Name	Address	Class	Section
Tuple 1 -->	1	ABC	Ghy	TDC	A
Tuple 2 -->	2	BHA	Delhi	HS	B
Tuple 3-->	3	JJGD	Mumbai	TDC	C

Relation Schema: A relation schema is a name of the relation with its attributes.

For example, Student(Roll_No., Name,Address,Class,Section) is a relation schema**.**

Degree of a relation or a relation schema: The number of attributes which in the relation or relation schema is called the degree of the relation or relation schema.

Student

Roll_No.	Name	Address	Class	Section
1	ABC	Ghy	TDC	A
2	BHA	Delhi	HS	B
3	JJGD	Mumbai	TDC	C

Student(Roll_No., Name,Address,Class,Section) is the relation schema for the above relation.

The degree of the Student relation schema or Student relation is 5.

Cardinality of a relation: The number of rows in a relation is called the cardinality of the relation.

Student

Roll_No.	Name	Address	Class	Section
1	ABC	Ghy	TDC	A
2	BHA	Delhi	HS	B
3	JJGD	Mumbai	TDC	C

The cardinality of this Student relation is 3.

Relation instance – A finite set of tuples in a relation schema is called a relation instance of that relation schema. A Relation instance can never have duplicate tuples.

Student

Roll_No.	Name	Address	Class	Section
1	ABC	Ghy	TDC	A
2	BHA	Delhi	HS	B
3	JJGD	Mumbai	TDC	C

Student_1

Roll_No.	Name	Address	Class	Section
1	ABC	Ghy	TDC	A
2	BHA	Delhi	HS	B

Here, Student_1 is a relation instance of Student relation.

Domain of attribute – The set of some pre-defined values and scopes of an attribute is called the domain of that attribute.

Relational constraints

Relational constraints refer to the rules or conditions which must be present for a valid relation.

The three main categories of relational constraints are:

1. Domain constraints
2. Key constraints
3. Integrity constraints

Domain Constraints

The domain constraint specifies that the values of an attribute must be appeared in the corresponding domain or of the appropriate data type.

Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

Key constraints

An attribute that can uniquely identify the tuples of a relation is called the key of the relation.

Different types of keys are-

Primary key:- An attribute or a set of attributes that can uniquely identifies the tuples in a relation is called a primary key.

Candidate key:- The attributes that have the property to become the primary key are called candidate key. i.e. all the attributes of a relation that can uniquely identify the tuples of the relation are called candidate key.

Superkey: - All the sets of attributes that can uniquely identify the tuples in a relation is called a superkey.

Alternate key:- A candidate key which is not a primary key is called an alternate key.

Integrity constraints
Two types of integrity constraints –

1. Entity Integrity

2. Referential Integrity

1. Entity Integrity:- The entity integrity states that no primary key value can be null.
2. Referential Integrity:- Referential integrity constraints is base on the concept of foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Relational Algebra

Relational algebra is a procedural query language, which is a set of operations performed on relations. It uses operators to perform queries. It includes the set theory operations and the relation database operations.

The relational database operations are-

1. Select
2. Project
3. Rename

The set theory operations are-

1. Union
2. Intersection
3. Difference
4. Cartesian Product

The operators for these operations can be either **unary** or **binary** depending upon the number of relations on which the operation is to be performed. The operators that perform operations on only one relation are called **unary** operator and the operators that perform operations on two relations are called binary operator. The relational algebra operators accept relations as their input and creates new relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

Select Operation

The select operation is denoted by σ. It selects tuples from a relation that satisfy the given. The syntax for select operation is -

$$\sigma_{<condition>}(\text{Relation\_Name})$$

The condition may use connectors like **and, or,** and **not** and relational operators like $=, \neq, \geq, <, >, \leq$.

For example −

$\sigma_{Eid>10}(\text{Employee})$

Selects tuples from Employee relation where Eid is greater than 10.

The select operator performs operation on only relation. So, it is a unary operator.

Project Operation

The project operation is denoted by π. This operation selects column(s) from a relation. The syntax for project operation is -

$\pi_{<\text{Attribute list}>}(\text{Relation\_Name})$

Duplicate rows are automatically eliminated, as relation is a set.

For example −

$\pi_{<\text{Eid,Ename}>}(\text{Employee})$

Selects the Eid and Ename from the Employee relation.

The Project operator performs operation on only relation. So, it is a unary operator.

Rename Operation

The rename operation is denoted by *ρ.* The rename operation allows to rename a relation. The syntax of rename operation is-

$\rho_{<\text{New\_Relation\_Name}>}$ (Existing_Relation_name)

For example,

$\rho_{<\text{Employee\_Under\_Project}>}$ (Employee)

Renames the relation Employee with Employee_Under_Project.

The Rename operator performs operation on only relation. So, it is a unary operator.

Union Operation

The union operation is denoted by ∪. This operation takes two relations as input and collects all the tuples from both the relations by removing the duplicate tuples. To perform the union operation the input relations must be union compatible. Two relations are said to be union compatible if they have the same number of attribute(i.e. the degree of the two input relations are same) and the domain of the corresponding attributes are same.

The syntax of the union operation is-

<Relation1> ∪ <Relation2>

For example,

Employee ∪ Employee_in_Project

Is possible if *Employee* and *Employee_in_Project* are union compatible and it collects all the tuples from *Employee* and *Employee_in_Project* by removing all the duplicate tuples.

Since the union operator takes two relations as input, so it is a binary operator.

Intersection

The intersection operator is denoted by ∩. This operation takes two relations and collects all the tuples which are present in both the relations. To perform the intersection operation the input relations must be union compatible.

The syntax of the intersection operation is-

<Relation1> ∩ <Relation2>

For example,

Employee ∩ Employee_in_Project

Is possible if *Employee* and *Employee_in_Project* are union compatible and it collects all the common tuples from both *Employee* and *Employee_in_Project* relations.

Since the intersection operator takes two relations as input, so it is a binary operator.

Set Difference

The difference operator is denoted by ─.The result of set difference operation is the set of tuples which are present in the first relation but not in the second relation. The syntax of this operation is-

<Relation1> ─ <Relation2>

For example,

Employee ─ Employee_in_Project

Finds all the tuples that are present in *Employee* but not in *Employee_in_Project.*

Cartesian Product

The Cartesian product operator is denoted by ×. It Combines tuples of the two relations into one. The syntax for this operation is-

<Relation1> x <Relation2>

For example,

Employee x Employee_in_Project

Combines each tuple of *Employee* with each tuple of *Employee_in_Project*