Relational Model, Relational Algebra and TRC

Codd's Rules

According to Dr Edgar F. Codd, there are twelve rules for ideal relational database:

Rule 1: Information Rule

Rule 2: Guaranteed Access Rule

Rule 3: Systematic Treatment of NULL Values

Rule 4: Active Online Catalog

Rule 5: Comprehensive Data Sub-Language Rule

Rule 6: View Updating Rule

Rule 7: High-Level Insert, Update, and Delete Rule

Rule 8: Physical Data Independence

Rule 9: Logical Data Independence

Rule 10: Integrity Independence

Rule 11: Distribution Independence

Rule 12: Non-Subversion Rule

RELATIONAL DATA MODEL

- It is a primary and simplest data model with following concepts:
- **Tables** – In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.
- **Tuple** – A single row of a table, which contains a single record for that relation is called a tuple.
- **Relation instance** – A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.
- **Relation schema** – A relation schema describes the relation name (table name), attributes, and their names.
- **Relation key** – Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.
- **Attribute domain** – Every attribute has some pre-defined value scope, known as attribute domain.

Constraints

- Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints.** There are three main integrity constraints –

Key Constraints

- There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key** for that relation. If there are more than one such minimal subsets, these are called ***candidate keys*.**
- Key constraints force that –
- In a relation with a key attribute, no two tuples can have identical values for key attributes.
- a key attribute can not have NULL values.

Domain Constraints

- Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Referential integrity Constraints

- Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.
- Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

Relational Algebra

Relational algebra is a **procedural** query language that works on relational model. The purpose of a query language is to retrieve data from database or perform various operations such as insert, update, delete on the data. When I say that relational algebra is a procedural query language, it means that it tells what data to be retrieved and how to be retrieved.

On the other hand relational calculus is a non-procedural query language, which means it tells what data to be retrieved but doesn’t tell how to retrieve it.

Types of operations in relational algebra

We have divided these operations in two categories:

1. Basic Operations
2. Derived Operations

Basic/Fundamental Operations:

1. Select (σ)
2. Project (∏)
3. Union (∪)
4. Set Difference (-)
5. Cartesian product (X)
6. Rename (ρ)

Derived Operations:

1. Natural Join (⋈)
2. Left, Right, Full outer join (⟕, ⟖, ⟗)
3. Intersection (∩)
4. Division (÷)

Select Operator (σ)

Select Operator is denoted by sigma (σ) and it is used to find the tuples (or rows) in a relation (or table) which satisfy the given condition.

If you understand little bit of SQL then you can think of it as a where clause in SQL, which is used for the same purpose.

Syntax of Select Operator (σ)

σ Condition/Predicate(Relation/Table name)

Project Operator (∏)

Project operator is denoted by ∏ symbol and it is used to select desired columns (or attributes) from a table (or relation).

Project operator in relational algebra is similar to the Select statement in SQL.

Syntax of Project Operator (∏)

∏ column_name1, column_name2, ..., column_nameN(table_name)

Syntax of Union Operator (∪)

table_name1 ∪ table_name2

Syntax of Intersection Operator (∩)

table_name1 ∩ table_name2

Set Difference (-)

- Set Difference is denoted by – symbol. Lets say we have two relations R1 and R2 and we want to select all those tuples(rows) that are present in Relation R1 but **not** present in Relation R2, this can be done using Set difference R1 – R2.
- **Syntax of Set Difference (-)**

  table_name1 - table_name2

Cartesian product (X)

- Cartesian Product is denoted by X symbol. Lets say we have two relations R1 and R2 then the Cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2. I know it sounds confusing but once we take an example of this, you will be able to understand this.
- **Syntax of Cartesian product (X) :** R1 X R2
- **Note:** The number of rows in the output will always be the cross product of number of rows in each table.

Rename (ρ)

- Rename (ρ) operation can be used to rename a relation or an attribute of a relation.
  Rename (ρ) Syntax: ρ(new_relation_name, old_relation_name)

Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms −

Tuple Relational Calculus (TRC)

Filtering variable ranges over tuples

Notation − {T | Condition}

Returns all tuples T that satisfies a condition.

For example −

```
{ T.name |  Author(T) AND T.article = 'database' }
```

Output − Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential (∃) and Universal Quantifiers (∀).

For example −

```
{ R| ∃T   ∈ Authors(T.article='database' AND R.name=T.name)}
```

Output − The above query will yield the same result as the previous one.

Domain Relational Calculus (DRC)

In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

Notation −

{ a_1, a_2, a_3, ..., a_n | P (a_1, a_2, a_3, ... ,a_n)}

Where a1, a2 are attributes and **P** stands for formulae built by inner attributes.

For example −

```
{< article, page, subject > |  ∈ TutorialsPoint ∧ subject =
'database'}
```

Output − Yields Article, Page, and Subject from the relation TutorialsPoint, where subject is database.

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

The expression power of Tuple Relation Calculus and Domain Relation Calculus is equivalent to Relational Algebra.