Relation Data Model
and Relational Database Constraints

في الطبقة المنتصف
(Representation data model
or
implementation Data model)

Relational data model

- Relational data model is based on relation.
- A relation is a mathematical concept based on the ideas of sets.
- Relational data model is the primary data model, which is used widely around the world for **data storage** and **processing**.
- This model is simple and it has all the properties and capabilities required to process data with storage efficiency.
- Relation هو الجدول

Concepts

Table- Tuple- relation(schema-instance- key)-attribute

- **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.

Concepts

- **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
  - Domain constraints
  - Referential integrity constraints قيود التكامل المرجعى

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**.
- Key constraints are also referred to as **Entity Constraints.**

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

- Relational database systems are expected to be equipped with **a query language** that can assist its users to query the database instances. There are two kinds of query languages – **relational algebra** and **relational calculus.**

Relational Algebra

- Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output.
- It uses **operators** to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output.
- **Relational algebra** is performed recursively on a relation and intermediate results are also considered relations.

Relational Algebra

- The fundamental operations of relational algebra are as follows −
  - Select
  - Project
  - Union
  - Set different
  - Cartesian product
  - Rename

Select Operation (σ)

- It selects tuples that satisfy the given predicate from a relation.

Notation – $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like – =, ≠, ≥, < , >, ≤.

For example –

$$\sigma_{subject = "database"}(Books)$$

Output – Selects tuples from books where subject is 'database'.

$$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$$

Output – Selects tuples from books where subject is 'database' and 'price' is 450.

$$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$$

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation – $\prod_{A_1, A_2, A_n} (r)$

Where A_1, A_2, A_n are attribute names of relation **r**.

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

For example –

$$\prod_{\text{subject, author}} (\text{Books})$$

Selects and projects columns named as subject and author from the relation Books.

Union Operation (U)

It performs binary union between two given relations and is defined as −

$$r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$$

Notation − r U s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$$\prod_{\text{author}} (\text{Books}) \cup \prod_{\text{author}} (\text{Articles})$$

Output − Projects the names of the authors who have either written a book or an article or both.

Set Difference (–)

The result of set difference operation is tuples, which are present in one relation but a
not in the second relation.

Notation – **r** – **s**

Finds all the tuples that are present in **r** but not in **s**.

$$\Pi_{author}(Books) - \Pi_{author}(Articles)$$

Output – Provides the name of authors who have written books but not articles.

Cartesian Product (X)

Combines information of two different relations into one.

Notation – r X s

Where **r** and **s** are relations and their output will be defined as –

r X s = { q t | q ∈ r and t ∈ s}

$$\sigma_{author = 'tutorialspoint'}(Books \times Articles)$$

Output – Yields a relation, which shows all the books and articles written by tutorialspoint.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** *ρ*.

Notation – $\rho_x(E)$

Where the result of expression **E** is saved with name of **x**.

Additional operations are –

- Set intersection
- Assignment
- Natural join

Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query 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 –

$\{ a_1, a_2, a_3, ..., a_n \mid P(a_1, a_2, a_3, ... ,a_n)\}$

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.