2. Relational Model

Objectives:

1. Structure of Relational Model
2. Codd's Rules
3. Parts of Relational Model
4. Relational Query Language
5. Relational Operations

Structure of Relational Model

Relational model was developed by E.F. Codd. The relational model is today the primary data model for commercial data processing applications. It attained its primary position because of its simplicity, which eases the job of the programmer, compared to earlier data models such as the network model or the hierarchical model.

Relational model is often described as having following aspects:

1. **Structural Aspect:** The data in database is perceived by the user as tables.
2. **Integrity Aspect:** Those tables satisfy certain integrity constrains.
3. **Manipulative aspect:** The operators available to the user for manipulating those tables.

A relational database consists of a collection of **tables**, each of which is assigned a unique name. A row in a table represents a *relationship* among a set of values. In the relational model the term **relation** is used to refer to a table, while the term **tuple** is used to refer to a row. Similarly, the term **attribute** refers to a column of a table. We use the term **relation instance** to refer to a specific instance of a relation, i.e., containing a specific set of rows.

Codd's Rules

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

Rule 0: The foundation rule:
For any system that is advertised as, or claimed to be, a relational data base management system, that system must be able to manage data bases entirely through its relational capabilities.

Rule 1: Information Rule
The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values
The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalogue
The structure description of the entire database must be stored in an online catalogue, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalogue which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule (Powerful Language Rule)
A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updation Rule
All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule (Relational Level Operation)
A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence
The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence
The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rules to apply.

Rule 10: Integrity Independence
A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence
The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule
Any row should obey the integrity & security constrains. If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

Parts of Relational Model

There are 3 parts of Relational Model:

1. **Structural:** Defines core of data & relationship.
    a. **Domain:** It is a set of values of same type.
    b. **Relational schema:** Overall design of relation (table).
    c. **Relation:** Relation of relational schema/algebra.
    $R(A_1, A_2, \ldots, A_n)$ is denoted by r(R), also known as list of tuples.
    $r(R) = \{t_1, t_2, \ldots, t_m\}$
    $t(S) = \{v_1, v_2, \ldots, v_n\}$ (list of values in tuples or **tuple shell**)
    d. **Degree:** Number of attributes in relation
    e. **Cardinality:** Number of tuples in relation

2. **Manipulative:** It defines how the data should be manipulated.

3. **Constraints:** It defines limits on model. It is set of rules & restrictions.

Constraints in the databases can be categorized into 3 main categories:

i. Constraints that are applied in the data model is called **Implicit constraints**.
    a. **Domain Constraints:** A set of permissive values that can be given to an attribute.
    b. **Key Constraints:** Key is a unique identity of a tuple/record. Key constraints allow to identify a record independently.
    c. **Entity integrity constraints:** There must be a primary key available to maintain consistency.
    d. **Referential Integrity constraints:** Values that appear in a relation should also appear in another relation.
    e. **Functional dependency constraints:** Establish a financial relation among 2 sets of attributes.

ii. Constraints that are directly applied in the schemas of the data model, by specifying them in the DDL (Data Definition Language). These are called as **schema-based constraints or Explicit constraints**.

iii. Constraints that cannot be directly applied in the schemas of the data model. We call these Application based or **semantic constraints**.

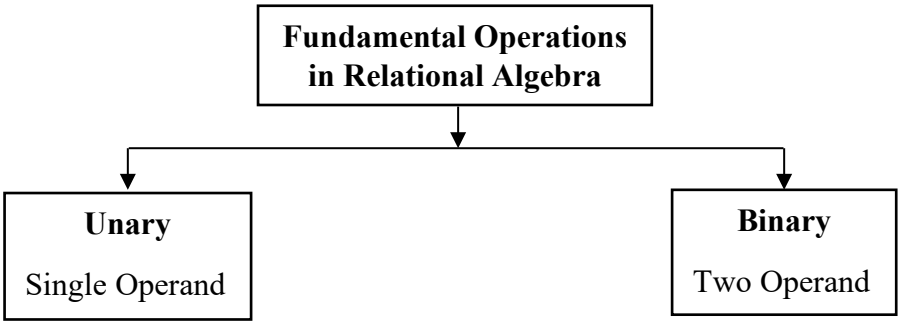
Relational Query Language

A **query language** is a language in which a user requests information from the database. These languages are usually on a level higher than that of a standard programming language. Query languages can be categorized as either procedural or nonprocedural.

In a **procedural language**, the user instructs the system to perform a sequence of operations on the database to compute the desired result. In a **nonprocedural language**, the user describes the desired information without giving a specific procedure for obtaining that information.

Relational Operations

We have relation as input and we get a new relation as output.
All procedural relational query languages provide a set of operations that can be applied to either a single relation or a pair of relations. These operations have the nice and desired property that their result is always a single relation. This property allows one to combine several of these operations in a modular way. Specifically, since the result of a relational query is itself a relation, relational operations can be applied to the results of queries as well as to the given set of relations.



Operator	Syntax	Example
Select	$\sigma_{conditions}(relation)$	$\sigma_{dept="comp"}(instructor)$
Project	$\Pi_{attributes}(relation)$	$\Pi_{name,dept}(instructor)$
Rename	$\rho_{new\_name/old\_name}(relation)$	$\rho_{dept/dept\_name}(instructor)$
Union	$R_1 \cup R_2$ **Rule-1:** Both relations must be of same arity **Rule-2:** Domain of i[th] element must be same	$\sigma(instructor) \cup \sigma(dept)$
Intersection	$R_1 \cap R_2$	$\sigma(instructor) \cap \sigma(dept)$
Set Difference	$R_1 - R_2$	$\sigma(instructor) - \sigma(dept)$
Cartesian Product	$R_1 \times R_2$	$\sigma(instructor) \times \sigma(dept)$
Natural Join	$R_1 \bowtie R_2$	$\sigma(instructor) \bowtie \sigma(dept)$
Semi Natural Join	$R_1 \ltimes R_2$ $R_1 \rtimes R_2$	$\sigma(instructor) \ltimes \sigma(dept)$ $\sigma(instructor) \rtimes \sigma(dept)$
Outer Join	$R_1 ⟗ R_2$ $R_1 ⟕ R_2$ $R_1 ⟖ R_2$	$\sigma(instructor) ⟗ \sigma(dept)$ $\sigma(instructor) ⟕ \sigma(dept)$ $\sigma(instructor) ⟖ \sigma(dept)$

Brief description will be given in Chapter-4 Formal Relational Query Languages.