Codd's Rule for Relational DBMS

E.F Codd was a Computer Scientist who invented the **Relational model** for Database management. Based on relational model, the **Relational database** was created. Codd proposed 13 rules popularly known as **Codd's 12 rules** to test DBMS's concept against his relational model. Codd's rule actualy define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS). Till now, there is hardly any commercial product that follows all the 13 Codd's rules. Even **Oracle** follows only eight and half(8.5) out of 13. The Codd's 12 rules are as follows.

Rule zero

This rule states that for a system to qualify as an **RDBMS**, it must be able to manage database entirely through the relational capabilities.

Rule 1: Information rule

All information(including metadata) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.

Rule 2: Guaranteed Access

Each unique piece of data (atomic value) should be accessible by: **Table Name + Primary Key (Row) + Attribute(column)**.

Rule 3: Systematic treatment of NULL

`Null` has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Also, Primary key must not be null, ever. Expression on `NULL` must give null.

Rule 4: Active Online Catalog

Database dictionary(catalog) is the structure description of the complete **Database** and it must be stored online. The Catalog must be governed by same rules as rest of the database. The same query language should be used on catalog as used to query database.

Rule 5: Powerful and Well-Structured Language

One well structured language must be there to provide all manners of access to the data stored in the database. Example: **SQL**, etc. If the database allows access to the data without the use of this language, then that is a violation.

Rule 6: View Updation Rule

All the view that are theoretically updatable should be updatable by the system as well.

Rule 7: Relational Level Operation

There must be Insert, Delete, Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

Rule 8: Physical Data Independence

The physical storage of data should not matter to the system. If say, some file supporting table is renamed or moved from one disk to another, it should not effect the application.

Rule 9: Logical Data Independence

If there is change in the logical structure(table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

Rule 10: Integrity Independence

The database should be able to enforce its own integrity rather than using other programs. Key and Check constraints, trigger etc, should be stored in Data Dictionary. This also make **RDBMS** independent of front-end.

Rule 11: Distribution Independence

A database should work properly regardless of its distribution across a network. Even if a database is geographically distributed, with data stored in pieces, the end user should get an impression that it is stored at the same place. This lays the foundation of **distributed database**.

Rule 12: Nonsubversion Rule

If low level access is allowed to a system it should not be able to subvert or bypass integrity rules to change the data. This can be achieved by some sort of looking or encryption.

Basic Relational DBMS Concepts

A **Relational Database management System**(RDBMS) is a database management system based on the relational model introduced by E.F Codd. In relational model, data is stored in **relations**(tables) and is represented in form of **tuples**(rows).

RDBMS is used to manage Relational database. **Relational database** is a collection of organized set of tables related to each other, and from which data can be accessed easily. Relational Database is the most commonly used database these days.

RDBMS: What is Table ?

In Relational database model, a **table** is a collection of data elements organised in terms of rows and columns. A table is also considered as a convenient representation of **relations**. But a table can have duplicate row of data while a true **relation** cannot have duplicate data. Table is the most simplest form of data storage. Below is an example of an Employee table.

ID	**Name**	**Age**	**Salary**
1	Adam	34	13000
2	Alex	28	15000
3	Stuart	20	18000
4	Ross	42	19020

RDBMS: What is a Tuple?

A single entry in a table is called a **Tuple** or **Record** or **Row**. A **tuple** in a table represents a set of related data. For example, the above **Employee** table has 4 tuples/records/rows.

Following is an example of single record or tuple.

1	Adam	34	13000

RDBMS: What is an Attribute?

A table consists of several records(row), each record can be broken down into several smaller parts of data known as **Attributes**. The above **Employee** table consist of four
attributes, **ID**, **Name**, **Age** and **Salary**.

Attribute Domain

When an attribute is defined in a relation(table), it is defined to hold only a certain type of values, which is known as **Attribute Domain**.

Hence, the attribute **Name** will hold the name of employee for every tuple. If we save employee's address there, it will be violation of the Relational database model.

Name
Adam
Alex
Stuart - 9/401, OC Street, Amsterdam
Ross

What is a Relation Schema?

A relation schema describes the structure of the relation, with the name of the relation(name of table), its attributes and their names and type.

What is a Relation Key?

A relation key is an attribute which can uniquely identify a particular tuple(row) in a relation(table).

Relational Integrity Constraints

Every relation in a relational database model should abide by or follow a few constraints to be a valid relation, these constraints are called as **Relational Integrity Constraints**.

The three main Integrity Constraints are:

1. Key Constraints
2. Domain Constraints
3. Referential integrity Constraints

Key Constraints

We store data in tables, to later access it whenever required. In every table one or more than one attributes together are used to fetch data from tables. The **Key Constraint** specifies that there should be such an attribute(column) in a relation(table), which can be used to fetch data for any tuple(row).

The Key attribute should never be **NULL** or same for two different row of data.

For example, in the **Employee** table we can use the attribute ID to fetch data for each of the employee. No value of ID is null and it is unique for every row, hence it can be our **Key attribute**.

Domain Constraint

Domain constraints refers to the rules defined for the values that can be stored for a certain attribute.

Like we explained above, we cannot store **Address** of employee in the column for **Name**.

Similarly, a mobile number cannot exceed 10 digits.

Referential Integrity Constraint

If a table reference to some data from another table, then that table and that data should be present for referential integrity constraint to hold true.

Examples of referential integrity constraint in the Customer/Order database of the Company: Customer(CustID, CustName) Order(OrderID, CustID, OrderDate).

The referential integrity constraint states that the customer ID (CustID) in the Order table must match a valid CustID in the Customer table. Most relational databases have declarative referential integrity. In other words, when the tables are created the referential integrity constraints are set up.