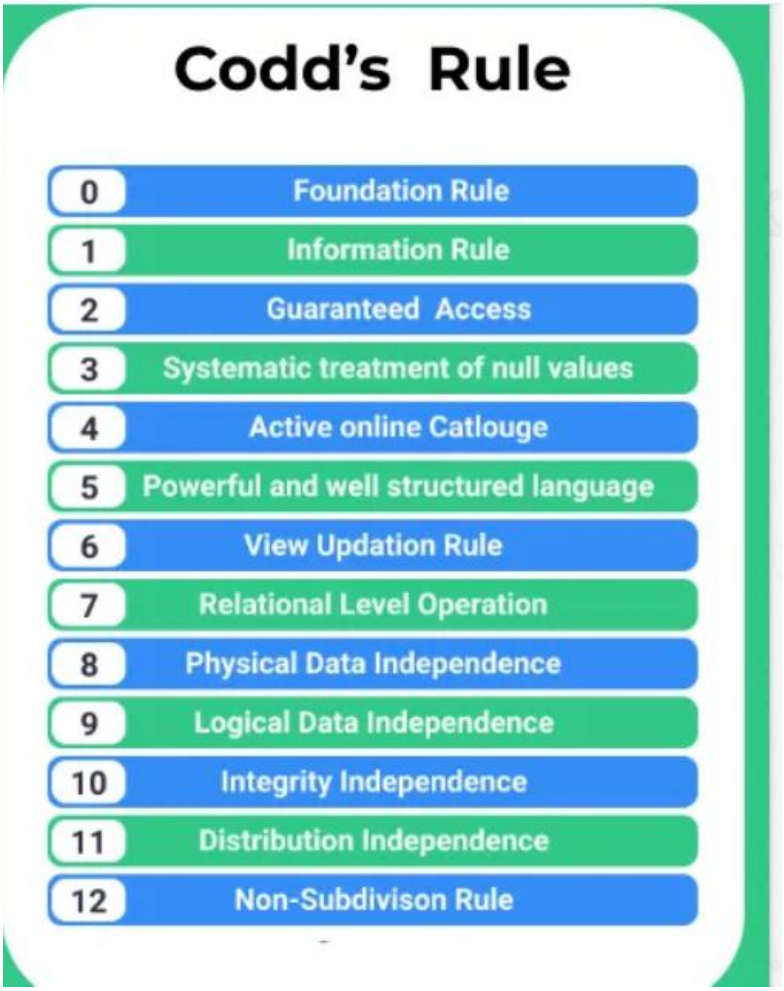
EF Codd's rules in DBMS

- EF Codd is a computer scientist who first outlined the relational model which now became the most popular and one and only database model.
- Codd Proposed 13 rules (listed from 0 to 12) popularly known as codd's 12 rules which are used in a relational database management system(RDBMS).
- Till now none of the commercial product has followed all the 13 rules even Oracle has followed 8.5 out of 13.



Rule 0: The Foundation Rule

- For any system to be considered a relational database management system (RDBMS), it must use relational capabilities to manage the database.
- Example: If you're using a system that can store data in tables (like a spreadsheet) and can use SQL (Structured Query Language) to manipulate and retrieve data, then it might be an RDBMS. If it doesn't use these relational methods, it's not an RDBMS.

Rule 1: The Information Rule

- All information in a relational database is represented explicitly at the logical level and in exactly one way: by values in tables.
- All the data must be stored in some cell of the table in the form of rows and columns. The Information Rule is fundamental to relational databases.
- It ensures that all the data stored in the database is represented in a clear and consistent way, using tables (also known as relations).
- Example:

EmployeeID	**EmpName**	**Age**
101	Alice	28
102	Bob	23

- **Data Representation:** Each row represents an employee, and each column represents an attribute of the employee, such as EmployeeID, Name, Department, and Salary.
- **Logical Level:** Users interact with the data through the table, using SQL queries.

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).
- The Guaranteed Access Rule ensures that every individual piece of data in a relational database can be uniquely identified and accessed.
- This means you can pinpoint any specific value using a clear and consistent method involving the table name, primary key, and column name.
- Example:

EmployeeID	**EmpName**	**Age**
101	Alice	28
102	Bob	23

Table Name: The table is named Employees.

Primary Key: EmployeeID uniquely identifies each row in the table.

Column Name: Each column represents a different attribute of the employee, such as Name ,age.

If you want to access the salary of the employee with EmployeeID102 (Bob), you can do so using:

Table Name: Employees

Primary Key: EmployeeID = 2

Column Name: Salary

Rule 3: Systematic Treatment of NULL Values

- NULL are systematically treated as missing information and are independent of data type.
- This rule ensures that NULL values, which represent missing or unknown data, are handled consistently and appropriately across the entire relational database system. NULL is a special marker used to indicate that a value is absent or unknown, and it is not the same as an empty string (e.g., '') or a zero (0).

Key Points:

- Distinct from Other Values: NULL is different from zero or an empty string. It specifically means "no value" or "unknown value."
- Independent of Data Type: NULL can be used in any column, regardless of the data type (e.g., integer, text, date).
- Example:

EmployeeID	**EmpName**	**Salary**
101	Alice	28000
102	Bob	NULL

- **Distinct from Other Values:** In this table, bob has a NULL in the Salary column, meaning his salary is unknown. Alice has a Valid value in salary column.

Rule 4: Active Online Catalog

- It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary.
- It authorizes users to access the database and implement a similar query language to access the database.
- The database description (metadata) is stored in an online catalog (data dictionary) that is accessible to users using the same query language used to access the database itself.

- This rule ensures that the metadata (information about the structure of the database) is stored within the database itself and can be accessed using the same language (typically SQL) that is used to access the actual data. This provides a consistent way to retrieve information about the database's structure, such as tables, columns, data types, and constraints.

- **Example:**
- Consider a database with two tables: Employees and Departments. The metadata about these tables (such as column names, data types, and constraints) is stored in system tables within the database.
- Accessing Metadata Using SQL
    - SELECT table_name FROM information_schema.tables WHERE table_schema = 'public';

| **table_name** |

|-------------|

| **Employees** |

| **Departments** |

Rule 5: The Comprehensive Data Sub-language

- This rule ensures that a relational database management system (RDBMS) provides a single, comprehensive language capable of performing all essential database operations.
- The language must allow users to define, manipulate, and control access to the data. In most RDBMSs, SQL (Structured Query Language) serves this purpose.
- Example Using SQL
    - Data Definition (DDL)

Creating a Table:

```
CREATE TABLE Employees (
EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    Department VARCHAR(100),
    Salary DECIMAL(10, 2)
```

);

This command creates an Employees table with four columns and sets EmployeeID as the primary key.

- Data Manipulation (DML)

Inserting Data:

INSERT INTO Employees (EmployeeID, Name, Department, Salary)

VALUES (1, 'Alice', 'HR', 60000);

Rule 6: View Updating Rule

- All the views of a database, which can theoretically be updated, must also be updatable by the system.
- A view is a virtual table that is based on the result of a SELECT query. It can be created to simplify complex queries, enhance security by restricting access to specific columns, or present data in a particular format.
- The View Updating Rule ensures that if a view logically allows for updates (insert, update, delete), then the database system should support these operations directly on the view, and the changes should reflect in the underlying base tables.

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

- The capability of handling a base relation or a derived relation (view) as a single operand applies not only to the retrieval of data but also to the insertion, update, and deletion of data.
- This rule mandates that a relational database management system (RDBMS) should allow users to perform high-level operations such as insert, update, and delete on sets of rows (relations) rather than just single rows. This means users can operate on entire tables or views with a single command, making data manipulation more efficient and expressive.

- Example:

  - High-Level Insert

    Inserting Multiple Rows:

    INSERT INTO Employees (EmployeeID, Name, Department, Salary) VALUES

    (1, 'Alice', 'HR', 60000),

    (2, 'Bob', 'IT', 70000),

(3, 'Charlie', 'Sales', 55000);

Rule 8: Physical Data Independence

- Changes to the physical storage of data (how data is stored on disk, access methods, etc.) should not require changes to the logical structure of the application or database.
- Physical data independence means that the way data is stored and accessed at a physical level can be changed without affecting the way data is viewed or interacted with at a logical level.
- This allows database administrators to optimize storage and access methods without disrupting the applications that use the data.
- If data is updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database.
- Let's break this down with a clear example:
    - Logical Operations (Database Level):
      INSERT INTO Employees (EmployeeID, Name, Department, Salary) VALUES (1, 'Alice', 'HR', 60000);
      INSERT INTO Employees (EmployeeID, Name, Department, Salary) VALUES (2, 'Bob', 'IT', 70000);
      INSERT INTO Employees (EmployeeID, Name, Department, Salary) VALUES (3, 'Charlie', 'Sales', 55000);
    - Physical Storage:
      employees_data.txt:
      1, Alice, HR, 60000
      2, Bob, IT, 70000
      3, Charlie, Sales, 55000
      The database administrator decides to change the storage format for better performance. The records are now stored in a binary file:
      employees_data.bin

Rule 9: Logical Data Independence

- Changes to the logical structure of the database (tables, columns, etc.) should not require changes to the applications that access the data. Logical data independence means that the logical schema (the structure of the database) can be changed without affecting the applications that interact with the database. This allows database administrators to modify the database schema to add new tables, columns, or relationships without needing to alter the existing applications.
- Example:
- Let's consider an Employees table:

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    Department VARCHAR(100),
    Salary DECIMAL(10, 2)
);
```

- **Adding a New Column**
  - ALTER TABLE Employees ADD COLUMN Email VARCHAR(100);
- new column Email is added to the Employees table.
- The structure of the table has changed.

- New applications can use the Email column, while existing applications remain unaffected.

Rule 10: Integrity Independence

- Integrity constraints (rules that ensure data integrity) must be definable in the database schema and not within application programs.
- Integrity independence means that the rules for maintaining data integrity should be part of the database itself rather than being enforced by the application code. This ensures that the integrity rules are consistently applied regardless of how the data is accessed or modified.

Rule 11: Distribution Independence

- The database should function properly and correctly regardless of whether the data is distributed across multiple locations or centralized in a single location.
- Distribution independence means that users should be able to access and interact with the data without needing to know or worry about whether the data is stored on a single server or spread across multiple servers in different locations. The database management system should handle data distribution transparently.

Rule 12: The Nonsubversion Rule

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