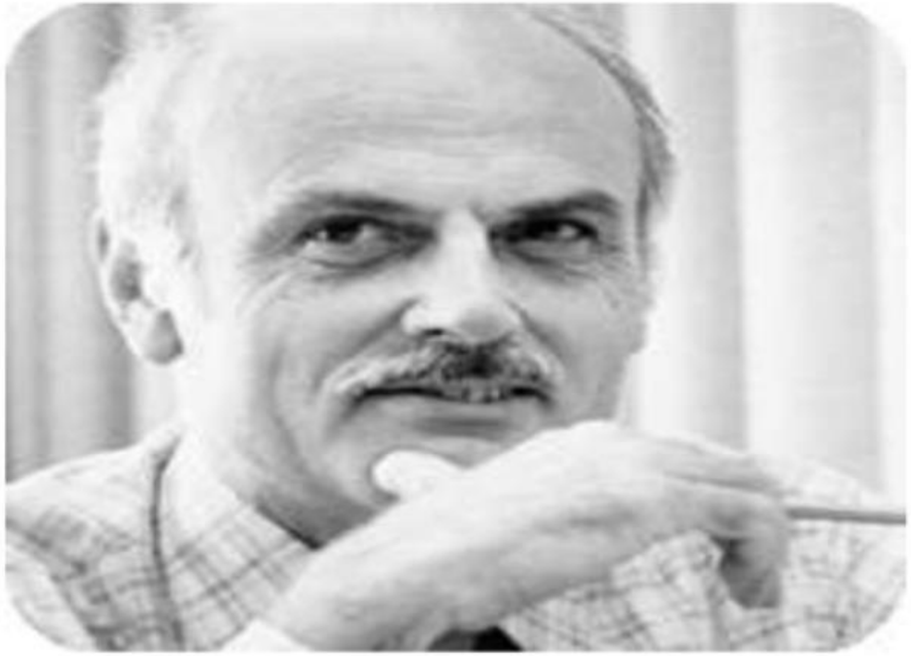
DATABASE SYSTEMS

CODD'S RULES

By Sana Faiz
Sana.faiz.muet83@gmail.com

CODD'S RULES

- Presented by Dr Edgar Frank Codd in 1979.
- Dr Edgar was a British-born American computer scientist and mathematician who devised the "**relational**" data model, which led to the creation of the relational **database,** a standard method of retrieving and storing computer data.
- 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.
- These rules are the pioneer rules for any relational database.



CODD'S RULES

1. The Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. That is , everything in a database must be stored in a table format.

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.

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.

4. Active Online Catalog

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

5. Comprehensive Data Sub-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.

6. View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

7. High-level Insert, Update and Delete Rule

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.

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.

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.

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.

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.

12. Non-subversion Rule

If a system has an interface that allows modification of records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

LOGICAL DATA INDEPENDENCE

- The ability to change the logical schema without changing the external schema or application programs is called as Logical Data Independence.

OR

- The ability to change the logical schema without having to change the external schema.

- **Example:** The addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

PHYSICAL DATA INDEPENDENCE

- The ability to change the physical schema without changing the logical schema is called as Physical Data Independence.

- Changes in the physical schema may include.

  - Using new storage devices.
  - Using different data structures.
  - Using different file organizations or storage structures.