Codd Rules

Codd rules were proposed by E.F. Codd which should be satisfied by relational model.

Every database has tables, and constraints cannot be referred to as a rational database system.

So, some rules define a database to be the correct RDBMS. These rules were developed by Dr. Edgar F. Codd (E.F. Codd) in 1985, who has vast research knowledge on the Relational Model of database Systems. Codd presents his 13 rules for a database to test the concept of DBMS against his relational model, and if a database follows the rule, it is called a true relational database (RDBMS). These 13 rules are popular in RDBMS, known as Codd's 12 rules.

1. **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.

2. **Information Rule:** Data stored in Relational model must be a value of some cell of a table.

3. **Guaranteed Access Rule:** Every data element must be accessible by table name, its primary key and name of attribute whose value is to be determined.

4. **Systematic Treatment of NULL values:** NULL value in database must only correspond to missing, unknown or not applicable values.

5. **Active Online Catalog:** Structure of database must be stored in an online catalog which can be queried by authorized users.

6. **Comprehensive Data Sub-language Rule:** A database should be accessible by a language supported for definition, manipulation and transaction management operation.

7. **View Updating Rule:** Different views created for various purposes should be automatically updatable by the system.

8. **High level insert, update and delete rule:** Relational Model should support insert, delete, update etc. operations at each level of relations. Also, set operations like Union, Intersection and minus should be supported.

9. **Physical data independence:** Any modification in the physical location of a table should not enforce modification at application level.

10. **Logical data independence:** Any modification in logical or conceptual schema of a table should not enforce modification at application level. For example, merging of two tables into one should not affect application accessing it which is difficult to achieve.

11. **Integrity Independence:** Integrity constraints modified at database level should not enforce modification at application level.

12. **Distribution Independence:** Distribution of data over various locations should not be visible to end-users.

13. **Non-Subversion Rule:** Low level access to data should not be able to bypass integrity rule to change data.

Relational Algebra

RELATIONAL ALGEBRA is a widely used procedural query language. It collects instances of relations as input and gives occurrences of relations as output. It uses various operations to perform this action. SQL Relational algebra query operations are performed recursively on a relation. The output of these operations is a new relation, which might be formed from one or more input relations.

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol: ρ)

Relational Algebra Operations From Set Theory

- UNION (υ)
- INTERSECTION ( ),
- DIFFERENCE (-)
- CARTESIAN PRODUCT (x)

Binary Relational Operations

- JOIN
- DIVISION

SELECT (σ)

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. Sigma(σ)Symbol denotes it. It is used as an expression to choose tuples which meet the selection condition. Select operator selects tuples that satisfy a given predicate.

$\sigma_p(r)$

σ is the predicate
r stands for relation which is the name of the table
p is prepositional logic

Example 1

$\sigma_{topic = "Database"}$ (Tutorials)

Output - Selects tuples from Tutorials where topic = 'Database'.

Example 2

$\sigma_{\text{topic = "Database" and author = "guru99"}}$(Tutorials)

Output - Selects tuples from Tutorials where the topic is 'Database' and 'author' is guru99.

Example 3

$\sigma_{\text{sales > 50000}}$ (Customers)

Output - Selects tuples from Customers where sales is greater than 50000

Projection(π)

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

This helps to extract the values of specified attributes to eliminates duplicate values. (pi) symbol is used to choose attributes from a relation. This operator helps you to keep specific columns from a relation and discards the other columns.

Example of Projection:

Consider the following table

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

Here, the projection of CustomerName and status will give

CustomerName	Status
Google	Active
Amazon	Active

$\Pi_{\text{CustomerName, Status}}$ (Customers)

Apple	Inactive
Alibaba	Active

Rename (ρ)

Rename is a unary operation used for renaming attributes of a relation.

ρ (a/b)R will rename the attribute 'b' of relation by 'a'.

Union operation (υ)

UNION is symbolized by ∪ symbol. It includes all tuples that are in tables A or in B. It also eliminates duplicate tuples. So, set A UNION set B would be expressed as:

The result <- A ∪ B

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

- R and S must be the same number of attributes.
- Attribute domains need to be compatible.
- Duplicate tuples should be automatically removed.

Example

Consider the following tables.

Table A			Table B	
column 1	column 2		column 1	column 2
1	1		1	1
1	2		1	3

A ∪ B gives

Table A □ B	
column 1	**column 2**
1	1

1	2
1	3

Set Difference (-)

- Symbol denotes it. The result of A - B, is a relation which includes all tuples that are in A but not in B.

- The attribute name of A has to match with the attribute name in B.
- The two-operand relations A and B should be either compatible or Union compatible.
- It should be defined relation consisting of the tuples that are in relation A, but not in B.

Example

A-B

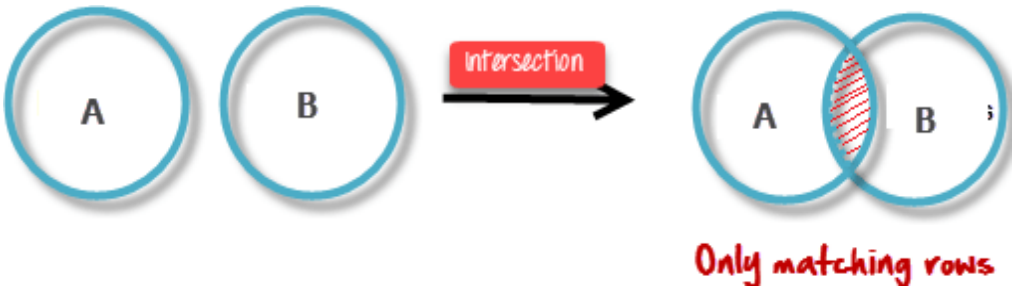
Table A - B	
column 1	**column 2**
1	2

Intersection

An intersection is defined by the symbol ∩

A ∩ B

Defines a relation consisting of a set of all tuple that are in both A and B.
However, A and Bmust be union-compatible.



Visual Definition of Intersection

Example:

A ∩ B

Table A ∩ B	
column 1	column 2
1	1

Cartesian Product(X) in DBMS

Cartesian Product in DBMS is an operation used to merge columns from two relations. Generally, a cartesian product is never a meaningful operation when it performs alone.
However, it becomes meaningful when it is followed by other operations. It is also calledCross Product or Cross Join.

Example – Cartesian product

$\sigma_{\text{column 2} = \text{'1'}}(A \times B)$

Output – The above example shows all rows from relation A and B whose column 2 hasvalue 1

σ column 2 = '1' (A X B)	
column 1	**column 2**
1	1
1	1

Transactional Control Commands

Transactional control commands are only used with the **DML Commands** such as - INSERT, UPDATE and DELETE only. They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

The COMMIT Command

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all the transactions to the database sincethe last COMMIT or ROLLBACK command.

The syntax for the COMMIT command is as follows.

```
COMMIT;
```

The ROLLBACK Command

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

The syntax for a ROLLBACK command is as follows −

```
ROLLBACK;
```

The SAVEPOINT Command

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain pointwithout rolling back the entire transaction.

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that pointwhenever required.

The syntax for a SAVEPOINT command is as shown below.

```
SAVEPOINT SAVEPOINT_NAME;
```

This command serves only in the creation of a SAVEPOINT among all the transactional statements.The ROLLBACK command is used to undo a group of transactions.

The syntax for rolling back to a SAVEPOINT is as shown below.

```
ROLLBACK TO SAVEPOINT_NAME;
```

Data Control Language

DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give "rights & permissions." Other permission controls parameters of the database system.

Examples of DCL commands:

Commands that come under DCL:

- Grant
- Revoke

Grant:

This command is use to give user access privileges to a database.

Syntax:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

For example:

```
GRANT SELECT ON Users TO'Tom'@'localhost;
```

Revoke:

It is useful to back permissions from the user.

Syntax:

```
REVOKE privilege_nameON object_nameFROM {user_name |PUBLIC |role_name}
```

For example:

```
REVOKE SELECT, UPDATE ON student FROM BCA, MCA;
```