Relational Model

1. **Relational Model**
   It organizes data in the form of **relations**, which you can understand as **tables**.
2. **Relational Database**
   A relational database is made up of many tables, and **each table has a unique name**.
3. **Rows in a Table**
   Every row in a table shows a **relationship between different values**.
   A table is basically a **collection of such rows**.
4. **Tuple**
   A tuple is simply **one row** in a table.
   It represents **one complete record**.
   Example: In a "Student" table, one student's data = one tuple.
5. **Columns**
   Columns represent the **attributes** (properties) of the table.
   Every attribute has a set of allowed values called its **domain**.
   Example: Age column → domain can be only positive numbers.
6. **Relation Schema**
   It defines the **structure** of the table.
   It includes:
   - Name of the table
   - All the columns/attributes
     Example: Student(Name, RollNo, Age)
7. **Examples of RDBMS**
   Databases that use the Relational Model are called **RDBMS**.
   Common ones: Oracle, IBM DB2, MySQL, MS Access.
8. **Degree of a Table**
   Degree = **number of columns** in a table.
   Example: Student(Name, RollNo, Age) → Degree = 3
9. **Cardinality**
   Cardinality = **number of rows** (tuples) in a table.
   Example: If the Student table has 50 students → Cardinality = 50

10. Relational Key

A relational key is a **set of attributes (columns)** that can **uniquely identify each row (tuple)** in a table.

11. Important Properties of a Table (Relation) in the Relational Model

1. **Each table must have a unique name.**
2. **Values must be atomic** — meaning each cell should contain a single value, not multiple values.
   (Example: A "Phone" column should store one number, not a list.)
3. **Each column name must be unique** inside a table.
4. **Every row must be unique** — no duplicate tuples.

5. **The order of rows and columns does not matter.**
   Swapping them does not change the meaning of the table.
6. **Tables must follow integrity constraints** to keep data correct and consistent across the database.

12. Types of Keys in Relational Model

1. Super Key (SK)

Any combination of attributes that can **uniquely identify a row**.
It may contain extra or unnecessary attributes.

Example:
In a Student table:

- {RollNo} → SK
- {RollNo, Name} → also SK (because RollNo alone is enough)

2. Candidate Key (CK)

A **minimal** super key.
It uniquely identifies a row **without any extra attributes**.

Important points:

- It must not have redundant attributes.
- **Its values cannot be NULL.**

Example:
Student table → RollNo, AadharNo can both be CKs.

3. Primary Key (PK)

One key selected from the set of candidate keys.
Usually the one with the **smallest number of attributes** and the most stable.

Example:
If both RollNo and AadharNo are CKs, we can choose RollNo as PK.

4. Alternate Key (AK)

All candidate keys **except** the primary key.

Example:
If RollNo is PK, then AadharNo becomes an Alternate Key.

5. Foreign Key (FK)

Used to create a link between **two tables**.

How it works:

- A table **r1** contains a column that refers to the **primary key of table r2**.
- This column is called the **foreign key**.

Naming:

- r1 → **Referencing (Child) table**
- r2 → **Referenced (Parent) table**

Example:

- Student table → StudentID (PK)
- Marks table → StudentID (FK referencing Student table)

FK ensures that data stays connected across tables.

6. Composite Key

A primary key that is created using **two or more attributes**.

Example:
(PaperCode, RollNo) → together form the PK.

7. Compound Key

A primary key that is formed using **two foreign keys**.

Example:
In a "StudentCourse" table:

- StudentID (FK)
- CourseID (FK)
  Together → PK (compound key)

8. Surrogate Key

A **synthetic or artificial primary key**, created automatically by the database.
Usually an integer, like: 1, 2, 3, 4...

Used when natural keys are long or complicated.

Example:
Auto-increment ID fields in MySQL.

13. Integrity Constraints

Integrity constraints are rules that make sure the database always remains **correct, valid, and consistent**, no matter what operations we perform.

1. All **CRUD operations** (Create, Read, Update, Delete) must follow certain rules so the database stays consistent.
2. These rules are introduced to **prevent accidental mistakes** that can corrupt or damage the data.
3. Integrity constraints ensure that only **valid and meaningful data** enters the database.

Types of Integrity Constraints

1. Domain Constraints

These rules **restrict the type of values** that can be stored in a column.

They ensure:

- The data type is correct (e.g., age should be a number).
- The values fall within an acceptable range.

Example:
If enrolment is allowed only for candidates whose birth year is **before 2002**, then:
BirthYear < 2002 → domain constraint.

This prevents entering invalid values like strings, or a birth year after 2002.

2. Entity Integrity Constraints

These rules ensure that each record in the table can be **uniquely identified**.

Main rule:

- Every table must have a **Primary Key**, and its value **cannot be NULL**.

This ensures:

- No record is left unidentified.
- No two rows have the same primary key.

5. Referential Constraints

Referential constraints are rules that ensure **relationships between two tables remain valid**.

1. They are defined **between two relations (tables)** and help keep data consistent across them.
2. The rule says:
   If a value appears in the **foreign key** of the referencing (child) table, that same value **must already exist** in the **primary key** of the referenced (parent) table.
3. So, if a foreign key refers to a primary key, then **every value of that foreign key must either:**
   - Appear in the parent table's primary key, **or**
   - Be **NULL** (if allowed).

4. This ensures that every foreign key value always has a matching parent record.

Simple Example:

- Parent Table: Department → DeptID (PK)
- Child Table: Employee → DeptID (FK)

If Employee table has DeptID = 10, then DeptID = 10 **must exist** in Department table.
Otherwise, the database will reject the entry.

6. Key Constraints

These are rules applied to specific columns to maintain correctness of the data.

1. NOT NULL

This constraint ensures a column **cannot have a NULL value**.
Every record must contain a value in this field.

Example:
Name column → NOT NULL (every person must have a name).

2. UNIQUE

This ensures all values in a column are **different from each other**.

Example:
Email column → UNIQUE (no two people can have the same email).

3. DEFAULT

Used to assign a **default value** to a column when no value is provided by the user.

Example:
Status column → DEFAULT 'Active'
If no value is entered, it automatically becomes “Active.”

4. CHECK

This constraint ensures that the value entered in a column **follows a specific condition**.

Example:
CHECK (Age > 18)
Only ages greater than 18 are allowed.

5. PRIMARY KEY

A primary key is a column (or set of columns) that **uniquely identifies each row**.
It must follow:

- NOT NULL
- UNIQUE

Example: RollNo in a Student table.

6. FOREIGN KEY

When two tables are related, one table contains a **foreign key** that refers to the primary key of another table.
This ensures that no action breaks the relationship between tables.

Example:
StudentID in the Marks table must match StudentID in the Student table.

This helps maintain **proper links** and prevents orphan records.