SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)
Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai
Accredited by NAAC-UGC with 'A++' Grade (Cycle III) &
Accredited by NBA (B.E - CSE, EEE, ECE, Mech & B.Tech.IT)
COIMBATORE-641 035, TAMIL NADU

UNIT II – Relational Model

Relational Data Model - keys, referential integrity and foreign keys, Relational Algebra - SQL fundamentals- Introduction, data definition in SQL, table, key and foreign key definitions, update behaviors-Views, Triggers, Joins, Constraints, Stored Procedure- Intermediate SQL-Advanced SQL features -Embedded SQL- Dynamic SQL

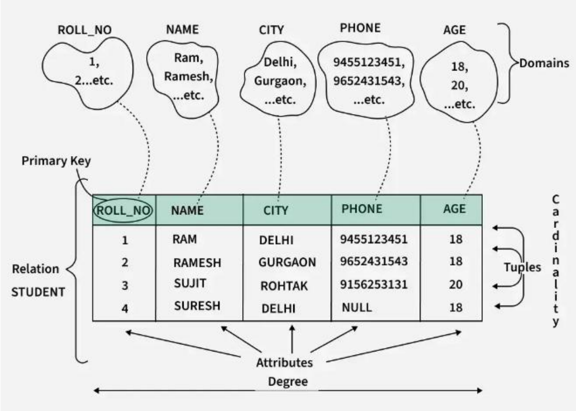
Relational Data Model

Overview

- The Relational Model represents data and their relationships through a collection of tables. Each table also known as a relation consists of rows and columns.
- Every column has a unique name and corresponds to a specific attribute, while each row contains a set of related data values representing a real-world entity or relationship.
- This model is part of the record-based models which structure data in fixed-format records each belonging to a particular type with a defined set of attributes.
- E.F. Codd introduced the Relational Model to organize data as relations or tables.
- After creating the conceptual design of a database using an ER diagram, this design must be transformed into a relational model which can then be implemented using relational database systems like Oracle SQL or MySQL.

What is relational Model?

The relational model represents how data is stored in Relational Databases. A relational database consists of a collection of tables each of which is assigned a unique name.



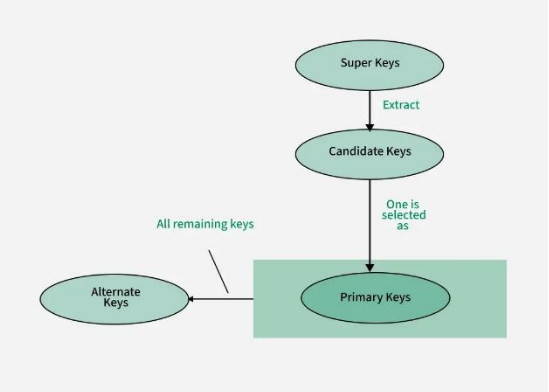
ROLL_NO	NAME	CITY	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	NULL	18

Key Terms

- **Attribute:** Attributes are the properties that define an entity. e.g. ROLL_NO, NAME, ADDRESS.
- **Relation Schema:** A relation schema defines the structure of the relation and represents the name of the relation with its attributes. e.g. STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT. If a schema has more than 1 relation it is called Relational Schema.
- **Tuple:** Each row in the relation is known as a tuple. The above relation contains 4 tuples one of which is shown as:

1	RAM	DELHI	9455123451	18

- **Relation Instance:** The set of tuples of a relation at a particular instance of time is called a relation instance. It can change whenever there is an insertion, deletion or update in the database.
- **Degree:** The number of attributes in the relation is known as the degree of the relation. The STUDENT relation defined above has degree 5.
- **Cardinality:** The number of tuples in a relation is known as cardinality. The STUDENT relation defined above has cardinality 4.
- **Column:** The column represents the set of values for a particular attribute. The column ROLL_NO is extracted from the relation STUDENT.
- **NULL Values:** The value which is not known or unavailable is called a NULL value. It is represented by NULL. e.g. PHONE of STUDENT having ROLL_NO 4 is NULL.
- **Relation Key:** These are basically the keys that are used to identify the rows uniquely or also help in identifying tables. These are of the following types:
    - Primary Key
    - Candidate Key
    - Super Key
    - Foreign Key
    - Alternate Key
    - Composite Key – (FK)



Relational Model Notation

- Relation schema R of degree n is denoted by by R(A1, A2, ...,An).
- Uppercase letters Q, R, S denote relation names.
- Lowercase letters q, r, s denote relation states.
- Letters t, u, v denote tuples.

- In general, the name of a relation schema such as STUDENT also indicates the current set of tuples in that relation.
- An attribute A can be qualified with the relation name R to which it belongs by using the dot notation R.A for example, STUDENT.Name or STUDENT.Age.
- An n-tuple t in a relation r(R) is represented as t=<v1,v2,…,vn> where vi is the value corresponding to the attribute Ai . The value vi for attribute Ai in tuple t can be accessed using t[Ai] or t.Ai.

Characteristics of the Relational Model

- **Data Representation**: Data is organized in tables (relations), with rows (tuples) representing records and columns (attributes) representing data fields.
- **Atomic Values**: Each attribute in a table contains atomic values, meaning no multi-valued or nested data is allowed in a single cell.
- **Unique Keys**: Every table has a primary key to uniquely identify each record, ensuring no duplicate rows.
- **Attribute Domain**: Each attribute has a defined domain, specifying the valid data types and constraints for the values it can hold.
- **Tuples as Rows**: Rows in a table, called tuples, represent individual records or instances of real-world entities or relationships.
- **Relation Schema**: A table's structure is defined by its schema, which specifies the table name, attributes, and their domains.
- **Data Independence**: The model ensures logical and physical data independence, allowing changes in the database schema without affecting the application layer.
- **Integrity Constraints**: The model enforces rules like:
  - **Domain constraints**: Attribute values must match the specified domain.
  - **Entity integrity**: No primary key can have NULL values.
  - **Referential integrity**: Foreign keys must match primary keys in the referenced table or be NULL.
- **Relational Operations**: Supports operations like **selection, projection, join, union, and intersection**, enabling powerful data retrieval manipulation.
- **Data Consistency**: Ensures data consistency through constraints, reducing redundancy and anomalies.
- **Set-Based Representation**: Tables in the relational model are treated as sets, and operations follow mathematical set theory principles.

Constraints in Relational Model

- While designing the Relational Model, we define some conditions which must hold for data present in the database are called Constraints.
- These constraints are checked before performing any operation (insertion, deletion, and updation) in the database.
- If there is a violation of any of the constraints, the operation will fail.

Domain Constraints

Domain Constraints ensure that the value of each attribute A in a tuple must be an atomic value derived from its specified domain, dom(A). Domains are defined by the data types associated with the attributes. Common data types include:

- **Numeric types:** Includes integers (short, regular, and long) for whole numbers and real numbers (float, double-precision) for decimal values, allowing precise calculations.
- **Character types:** Consists of fixed-length (CHAR) and variable-length (VARCHAR, TEXT) strings for storing text data of various sizes.
- **Boolean values:** Stores true or false values, often used for flags or conditional checks in databases.
- **Specialized types:** Includes types for date (DATE), time (TIME), timestamp (TIMESTAMP), and money (MONEY), used for precise handling of time-related and financial data.

Key Integrity

Every relation in the database should have at least one set of attributes that defines a tuple uniquely. Those set of attributes is called keys. e.g.; ROLL_NO in STUDENT is key. No two students can have the same roll number. So, a key has two properties:

- It should be unique for all tuples.
- It can't have NULL values.

Referential Integrity Constraints

When one attribute of a relation can only take values from another attribute of the same relation or any other relation, it is called referential integrity.

Relation which is referencing another relation

Table: Student

Roll_No	Name	Address	Phone	Age	Branch_Code
1	RAM	DELHI	9455123451	18	CS

Roll_No	Name	Address	Phone	Age	Branch_Code
2	RAMESH	GURGAON	9652431543	18	CS
3	SUJIT	ROHTAK	9156253131	20	ECE
4	SURESH	DELHI	1234567890	18	IT

Table :Branch

Branch_Code	Branch_Name
CS	Computer Science
IT	Information Technology
ECE	Electronics And Communication Engineering
CV	Civil Engineering

Anomalies in the Relational Model

An anomaly is an irregularity or something which deviates from the expected or normal state. When designing databases, we identify three types of anomalies: **Insert**, **Update**, and **Delete**.

Insertion Anomaly in Referencing Relation

We can't insert a row in REFERENCING RELATION if referencing attribute's value is not present in the referenced attribute value. e.g.; Insertion of a student with BRANCH_CODE 'ME' in STUDENT relation will result in an error because 'ME' is not present in BRANCH_CODE of BRANCH.

Deletion/ Updation Anomaly in Referenced Relation:

We can't delete or update a row from REFERENCED RELATION if the value of REFERENCED ATTRIBUTE is used in the value of REFERENCING ATTRIBUTE. e.g. if we try to delete a tuple from BRANCH having BRANCH_CODE 'CS', it will result in an error because 'CS' is referenced by BRANCH_CODE of STUDENT, but if we try to delete the row from BRANCH with BRANCH_CODE CV, it will be deleted as the value is not been used by referencing relation. It can be handled by the following method:

On Delete Cascade

It will delete the tuples from REFERENCING RELATION if the value used by REFERENCING ATTRIBUTE is deleted from REFERENCED RELATION. e.g.; if we delete a row from BRANCH with BRANCH_CODE 'CS', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be deleted.

On Update Cascade

It will update the REFERENCING ATTRIBUTE in REFERENCING RELATION if the attribute value used by REFERENCING ATTRIBUTE is updated in REFERENCED RELATION. e.g., if we update a row from BRANCH with BRANCH_CODE 'CS' to 'CSE', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be updated with BRANCH_CODE 'CSE'.