

Fundamentals of Database Systems



Chapter 2: Relational Data Models



Cont...

- A relational model is a model that is perceived by the user as **table**.
- **Relation:** Two dimensional table
 - Stores information or data in the form of tables (rows and columns)
 - A **row** of the table is called tuple(equivalent to record)
 - A **column** of a table is called attribute(equivalent to fields)
- Data value is the value of the Attribute

Cont...

- The tables seem to be independent but are related some how.
- No physical consideration of the storage is required by the user
- Many tables are merged together to come up with a new virtual view of the relationship

Alternative terminologies		
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Relational Data Model

- The relational model represents the database as a collection of *relations*.
- **Properties of Relational Databases**
 - A relation has distinct name
 - Each tuple in a relation must be unique
 - All tables are **LOGICAL ENTITIES**
 - Each cell of a relation contains exactly one atomic value.
 - Each column/field/attribute has a distinct name.
- Relational database is the collection of tables
 - Each entity in one table
 - Attributes are fields (columns) in table
- Entries with repeating groups are said to be un-normalized
- All values in a column represent the same attribute and have the same data format

Building Blocks of the Relational Data Model

- **Entities**
- **Attributes**
- **Relationships**
- **Constraints**
- **Entities:** Real world physical or logical objects
 - The **ENTITIES** (persons, places, things etc.) which the organization has to deal with.
 - Entity names → singular noun
- **Attributes:**
 - properties used to describe each Entity/real world object.
 - **Attribute name** → explanatory words/phrases
 - **The domain** → attribute values are taken from

Types of Attributes

- Simple (atomic) Vs Composite
 - **Simple** : contains a single value (not divided into sub parts)
 - E.g. Age, gender
 - **Composite**: Divided into sub parts (composed of other attributes)
 - E.g. Name, address
- Single-valued Vs multi-valued
 - **Single-valued** : have only single value(the value may change but has only one value at one time)
 - E.g. Name, Sex, Id.
 - **Multi-Valued**: have more than one value
 - E.g. Address, dependent-name
 - Person may have several college degrees

- **Stored vs. Derived Attribute**
 - **Stored** : not possible to derive or compute
 - E.g. Name, Address
 - **Derived**: The value may be derived (computed) from the values of other attributes.
 - E.g. Age (current year – year of birth)
 - Length of employment (current date- start date)
 - Profit (earning-cost), G.P.A (grade point/credit hours)
- **Null Values**
 - NULL applies to attributes which are not applicable or which do not have values.
 - You may enter the value NA (meaning not applicable)
 - Value of a key attribute can not be null.
- **Default value** - assumed value if no explicit value

- **Relationship:** the association between Entities
 - **Degree of a Relationship**
 - The number of entities participating in a relationship is called the **DEGREE** of the relationship.
 - Basic Degrees of relationship:
 - **UNARY/RECURSIVE RELATIONSHIP:** *Tuples/records of a Single entity are related withy each other.*
 - **BINARY RELATIONSHIPS:** *Tuples/records of two entities are associated in a relationship*
 - **TERNARY RELATIONSHIP:** *Tuples/records of three different entities are associated*
 - And a generalized one:
 - **N-ARY RELATIONSHIP:** *Tuples from arbitrary number of entity sets are participating in a relationship*

- **Cardinality of a Relationship**
- The number of instances participating or associated with a single instance from an entity in a relationship is called the **CARDINALITY** of the relationship.
- The major cardinalities of a relationship are:
 - ONE-TO-ONE: one tuple is associated with only one other tuple.
 - E.g. Building – Location → as a single building will be located in a single location and as a single location will only accommodate a single Building.
 - ONE-TO-MANY, one tuple can be associated with many other tuples, but not the reverse.
 - E.g. Department-Student → as one department can have multiple students.
 - MANY-TO-ONE, many tuples are associated with one tuple but not the reverse.
 - E.g. Employee – Department: as many employees belong to a single department.
 - MANY-TO-MANY: one tuple is associated with many other tuples and from the other side, with a different role name one tuple will be associated with many tuples
 - E.g. Student – Course → as a student can take many courses and a single course can be attended by many students.

- Constraints
 - rules that should be obeyed while manipulating the data.
- Key constraints:
 - **Super Key**: an attribute or set of attributes that uniquely identifies a tuple within a relation.
 - **Candidate Key**: a super key such that no proper subset of that collection is a Super Key within the relation.
 - A candidate key has two properties:
 - **Uniqueness**
 - **Irreducibility**
 - If a super key is having only one attribute, it is automatically a Candidate key.
 - If a candidate key consists of more than one attribute it is called Composite Key.

- **Primary Key:**
 - the candidate key that is selected to identify tuples uniquely within the relation.
- **Foreign Key:**
 - an attribute, or set of attributes, within one relation that matches the candidate key of some relation.
 - A foreign key is a link between different relations to create a view or an unnamed relation
- **Relational Integrity Rules**
 - **Domain Integrity:** No value of the attribute should be beyond the allowable limits
 - **Entity Integrity:** In a base relation, no attribute of a Primary Key can assume a value of NULL
 - **Referential Integrity:** If a Foreign Key exists in a relation, either the Foreign Key value must match a Candidate Key value in its home relation or the Foreign Key value must be NULL
 - **Enterprise Integrity:** Additional rules specified by the users or database administrators of a database are incorporated

- **Relational Views**
 - Relations are perceived as a Table from the users' perspective.
 - The two categories of Relations:
 - Named and Unnamed Relations.
 - The basic difference is on how the relation is created, used and updated
- **Base Relation/Named Relations**
 - corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.
- **View (Unnamed Relation)**
 - Result of relational operations operating on the base relations to produce another virtual relation.(does not actually exist)
 - is **virtually derived relation** that does not exist in the database
 - Virtual table can be created from single/different relations by extracting some attributes & records with/without conditions.

- **Purpose of a view**

- Hides unnecessary information from users.
- Provide powerful flexibility and security.
- Provide customized view of the database for users
- A view of one base relation can be updated.
- Update on views derived from various relations is not allowed since it may violate the integrity of the database.
- Update on view with aggregation and summary is not allowed. Since aggregation and summary results are computed from a base relation and does not exist actually.

Schemas and Instances and Database State

- Two basic components of Relational database:
 1. The Definition of the Relation/Table
 - The data definition is the Schema or the skeleton of the database
 2. The actual data stored in each table.
 - The Relations with some information at some point in time is the Instance or the flesh of the database.

- **Schemas**
 - Schema describes how data is to be structured & defined at Design time (also called "metadata") or
 - May be during system maintenance time etc..
- **Database Schema (Intension):**
 - Specifies name of relation & collection of the attributes (or Name of attributes).
 - refer to a description of database (or intention)
 - specified during database design
 - should not be changed unless during maintenance
 - **Schema Diagrams**
 - convention to display some aspect of a schema visually
 - **Schema Construct**
 - refers to each object in the schema (e.g. STUDENT), STUNEDT (F_name, L-Name, Id,Year,Dept, Sex)

- **Instance:**

- collection of data in the database at a particular point of time (snap-shot).
- Refers to the actual data in the database at a specific point in time
- State of database is changed any time we add, delete or update an item.
- ***Valid state***: the state that satisfies the structure and constraints specified in the schema and is enforced by DBMS
- Actual data of the database at some point in time that changes rapidly