

---

## Chapter Two

# Relational Data Model

### **Important terms:**

**Relation:** a table with rows and columns

**Attribute:** a named column of a relation

**Domain:** a set of allowable values for one or more attributes

**Tuple:** a row of a relation

**Degree:** the degree of a relation is the number of attributes it contains

*Unary relation, Binary relation, Ternary relation, N-ary relation*

**Cardinality:** of a relation is the number of tuples the relation has

**Relational Database:** a collection of normalized relations with distinct relation names.

**Relation Schema:** a named relation defined by a set of attribute-domain name pair

Let  $A_1, A_2, \dots, A_n$  be attributes with domain  $D_1, D_2, \dots, D_n$ .

Then the sets  $\{A_1:D_1, A_2:D_2, \dots, A_n:D_n\}$  is a Relation Schema. A relation  $R$ , defined by a relation schema  $S$ , is a set of mappings from attribute names to their corresponding domains. Thus a relation is a set of  $n$ -tuples of the form  $(A_1:d_1, A_2:d_2, \dots, A_n:d_n)$  where  $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$ .

Eg.

Student (studentId char(10), studentName char(50), DOB date) is a **relation schema** for the student entity in SQL

**Relational Database schema:** a set of relation schema each with distinct names.

Suppose  $R_1, R_2, \dots, R_n$  is the set of relation schema in a relational database then the relational database schema ( $R$ ) can be stated as

$R = \{ R_1, R_2, \dots, R_n \}$

---

---

## Properties of Relational Databases

- A relation has a name that is distinct from all other relation names in the relational schema.
- Each tuple in a relation must be unique
- All tables are **LOGICAL ENTITIES**
- Each cell of a relation contains exactly one atomic (single) value.
- Each column (field or attribute) has a distinct name.
- The values of an attribute are all from the same domain.
- A table is either a BASE TABLES (Named Relations) or VIEWS (Unnamed Relations)
- Only Base Tables are physically stored
- VIEWS are derived from BASE TABLES with SQL statements like: [SELECT .. FROM .. WHERE .. ORDER BY]
- Relational database is the collection of tables
  - Each entity in one table
  - Attributes are fields (columns) in table
- Order of rows theoretically ( but practically has impact on performance) and columns is immaterial
- 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**

The building blocks of the relational data model are:

- **Entities:** real world physical or logical object
- **Attributes:** properties used to describe each Entity or real world object.
- **Relationship:** the association between Entities
- **Constraints:** rules that should be obeyed while manipulating the data.

1. The **ENTITIES** (persons, places, things etc.) which the organization has to deal with. Relations can also describe relationships

The name given to an entity should always be a singular noun descriptive of each item to be stored in it. E.g. : student *NOT* students.

Every relation has a schema, which describes the columns, or fields the relation itself corresponds to our familiar notion of a table:

A relation is a collection of *tuples*, each of which contains values for a fixed number of *attributes*

- Existence Dependency: the dependence of an entity on the existence of one or more entities.
- Weak entity : an entity that can not exist without the entity with which it has a relationship – it is indicated by a **double rectangle**

2. The **ATTRIBUTES** - the items of information which characterize and describe these entities.

Attributes are pieces of information ABOUT entities. The analysis must of course identify those which are actually relevant to the proposed application. Attributes will give rise to recorded items of data in the database

At this level we need to know such things as:

- **Attribute name** (be explanatory words or phrases)
  - **The domain** from which attribute values are taken (A DOMAIN is a set of values from which attribute values may be taken.) Each attribute has values taken from a *domain*. For example, the
-

---

domain of Name is string and that for salary is real. However these are not shown on **E-R models**

- Whether the attribute is part of the **entity identifier** (attributes which just describe an entity and those which help to identify it uniquely)
- Whether it is **permanent or time-varying** (which attributes may change their values over time)
- Whether it is **required or optional** for the entity (whose values will sometimes be unknown or irrelevant)

## Types of Attributes

### (1) Simple (atomic) Vs Composite attributes

- **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

### (2) Single-valued Vs multi-valued attributes

- **Single-valued** : have only single value (the value may change but has only one value at one time)  
E.g. Name, Sex, Id. No. color\_of\_eyes
- **Multi-Valued**: have more than one value  
E.g. Address, dependent-name  
Person may have several college degrees

### (3) 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)

### (4) 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

## Entity versus Attributes

When designing the conceptual specification of the database, one should pay attention to the distinction between an Entity and an Attribute.

- Consider designing a database of employees for an organization:
- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
  - If we have several addresses per employee, *address* must be an entity (*attributes cannot be set-valued/multi valued*)
- If the structure (city, Woreda, Kebele, etc) is important, e.g. want to retrieve employees in a given city, address must be modeled as an entity (*attribute values are atomic*)

---

3. The **RELATIONSHIPS** between entities which exist and must be taken into account when processing information. In any business processing one object may be associated with another object due to some event. Such kind of association is what we call a **RELATIONSHIP** between entity objects.

- One external event or process may affect several related entities.
- Related entities require setting of **LINKS** from one part of the database to another.
- A relationship should be named by a word or phrase which explains its function
- Role names are different from the names of entities forming the relationship: one entity may take on many roles, the same role may be played by different entities
- For each **RELATIONSHIP**, one can talk about the Number of Entities and the Number of Tuples participating in the association. These two concepts are called **DEGREE** and **CARDINALITY** of a relationship respectively.

## Degree of a Relationship

- An important point about a relationship is how many entities participate in it. The number of entities participating in a relationship is called the **DEGREE** of the relationship.

Among the Degrees of relationship, the following are the basic:

- **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

- Another important concept about relationship is the number of instances/tuples that can be associated with a single instance from one entity in a single 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.

However, the degree and cardinality of a *relation* are different from degree and cardinality of a *relationship*.

---

---

## Key constraints

If tuples are need to be unique in the database, and then we need to make each tuple distinct. To do this we need to have relational keys that uniquely identify each record.

**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:

1. **Uniqueness**
2. **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.

The entire set of attributes in a relation can be considered as a primary case in a worst case.

**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 Constraints/Integrity Rules

- **Relational Integrity**
    - **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. Actually, there are two kinds of relation in relational database. The two categories or types of Relations are Named and Unnamed Relations. The basic difference is on how the relation is created, used and updated:

1. **Base Relation**

A *Named Relation* corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.

2. **View (Unnamed Relation)**

A View is the dynamic result of one or more relational operations operating on the base relations to produce another virtual relation that does not actually exist as presented. So a view is **virtually derived relation** that does not necessarily exist in the database but can be produced upon request by a particular user at the time of request. The virtual table or relation can be created from single or different relations by extracting some attributes and records with or without conditions.

## Purpose of a view

- Hides unnecessary information from users: since only part of the base relation (Some collection of attributes, not necessarily all) are to be included in the virtual table.
  - Provide powerful flexibility and security: since unnecessary information will be hidden from the user there will be some sort of data security.
  - Provide customized view of the database for users: each user is going to be interfaced with their own preferred data set and format by making use of the Views.
  - 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

When a database is designed using a Relational data model, all the data is represented in a form of a table. In such definitions and representation, there are two basic components of the database. The two components are the definition of the Relation or the Table and the actual data stored in each table. The data definition is what we call the Schema or the skeleton of the database and 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 setup/Design time (also called "metadata")
- Since it is used during the database development phase, there is rare tendency of changing the schema unless there is a need for system maintenance which demands change to the definition of a relation.
- **Database Schema (Intension):** specifies name of relation and the collection of the attributes (specifically the 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)  
E.g.: STUDENT (FName, LName, Id, Year, Dept, Sex)

---

# Instances

- **Instance:** is the collection of data in the database at a particular point of time (snap-shot).
    - Also called **State or Snap Shot or Extension of the** database
    - 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
  - Since Instance is actual data of database at some point in time, changes rapidly
  - To define a new database, we specify its database schema to the DBMS (database is empty)
  - database is initialized when we first load it with data
-