Overview of DBMS

Uma Seshadri

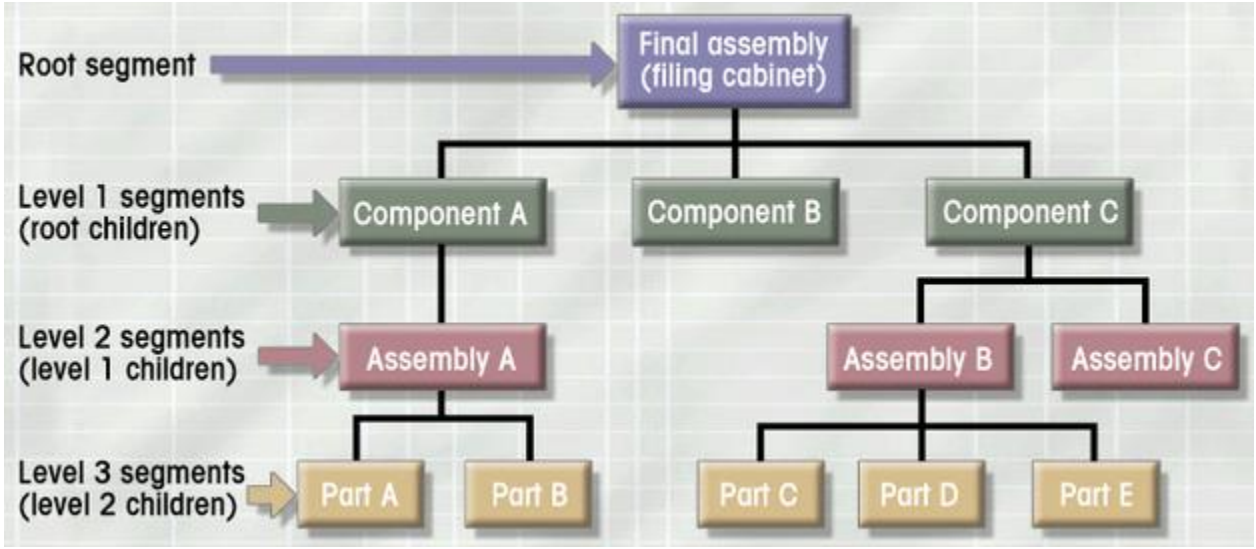
Outline

- Types of Databases overview
- Data Models
- Relational Databases

Hierarchical Database

- Background
  - Developed to manage large amount of data for complex manufacturing projects
  - e.g., Information Management System (IMS)
    - IBM-Rockwell joint venture
    - clustered related data together
    - hierarchically associated data clusters using pointers
- Hierarchical Database Model
  - Assumes data relationships are hierarchical
    - One-to-Many (1:M) relationships
      - Each parent can have many children
      - Each child has only one parent
  - Logically represented by an upside down tree

Hierarchical Database: Example



Database Systems: Design, Implementation, & Management: Rob & Coronel

Hierarchical Database: Pros & Cons

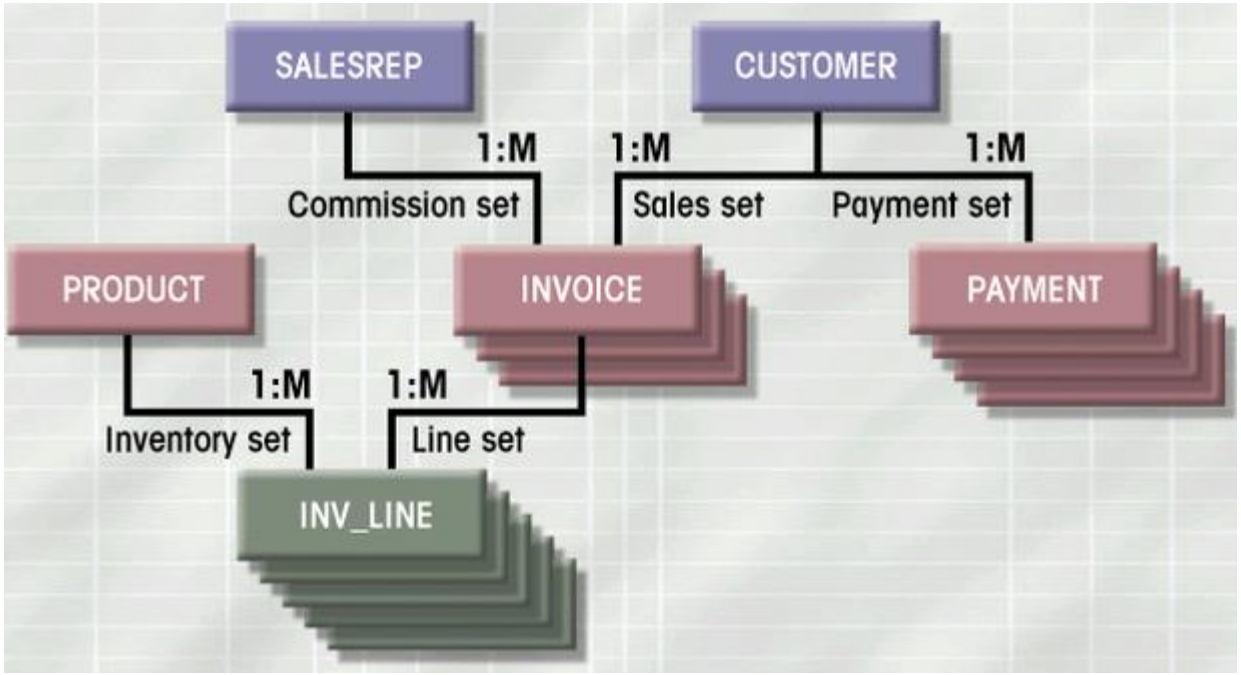
- Advantages
  - Conceptual simplicity
    - groups of data could be related to each other
    - related data could be viewed together
  - Centralization of data
    - reduced redundancy and promoted consistency
- Disadvantages
  - Limited representation of data relationships
    - did not allow Many-to-Many (M:N) relations
  - Complex implementation
    - required in-depth knowledge of physical data storage
  - Structural Dependence
    - data access requires physical storage path
  - Lack of Standards
    - limited portability

Network Database

- Objectives
  - Represent more complex data relationships
  - Improve database performance
  - Impose a database standard

- Network Database Model
  - Similar to Hierarchical Model
    - Records linked by pointers
  - Composed of sets
    - Each set consists of owner (parent) and member (child)
  - Many-to-Many (M:N) relationships representation
    - Each owner can have multiple members (1:M)
    - A member may have several owners

Network Database: Example



Database Systems: Design, Implementation, & Management: Rob & Coronel

Network Database: Pros & Cons

- Advantages
  - More data relationship types
  - More efficient and flexible data access
    - “network” vs. “tree” path traversal
  - Conformance to standards
    - enhanced database administration and portability
- Disadvantages
  - System complexity
    - require familiarity with the internal structure for data access
  - Lack of structural independence
    - small structural changes require significant program changes

Relational Database

- Problems with legacy database systems
  - Required excessive effort to maintain
    - Data manipulation (programs) too dependent on physical file structure
  - Hard to manipulate by end-users
    - No capacity for ad-hoc query (must rely on DB programmers).
- Evolution in Data Organization
  - E. F. Codd's Relational Model proposal
    - Separated the notion of physical representation (machine-view) from logical representation (human-view)
    - Considered ingenious but computationally impractical in 1970

Relational Database (contd.)

- Relational Database Model
  - Dominant database model of today
  - Eliminated pointers and used tables to represent data
  - Tables
    - flexible logical structure for data representation
    - a series of row/column intersections
    - related by sharing common entity characteristic(s)

Relational Database: Example

- Provides a logical "human-level" view of the data and associations among groups of data (i.e., tables)

Customer_ID	Customer_Account	Agent_ID
1224	4556	23
1225	4558	25

Agent_ID	Last_Name	First_Name	Phone
23	Sturm	David	334-5678
25	Long	Kyle	556-3421

Customer_ID	Last_Name	First_Name	Phone	Account_Balance
1224	Vira	Dyne	678-9987	1223.95
1225	Davies	Tricia	556-3342	234.25

Relational Database: Pros & Cons

- Advantages
  - Structural independence
    - Separation of database design and physical data storage/access
    - Easier database design, implementation, management, and use
  - Ad hoc query capability with Structured Query Language (SQL)
    - SQL translates user queries to codes
- Disadvantages
  - Substantial hardware and system software overhead
    - more complex system
  - Poor design and implementation is made easy
    - ease-of-use allows careless use of RDBMS

Entity Relationship Model

- Peter Chen's Landmark Paper in 1976
  - "The Relationship Model: Toward a Unified View of Data"
  - Graphical representation of entities and their relationships

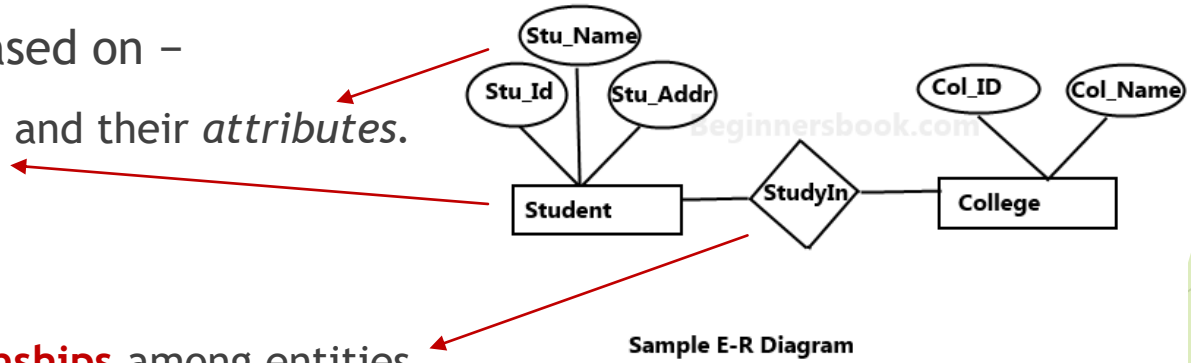
- Entity Relationship (ER) Model
  - Based on Entity, Attributes & Relationships
    - Entity is a thing about which data are to be collected and store
    - Attributes are characteristics of the entity
    - Relationships describe an associations between entities
      - i.e. 1:M, M:N, 1:1
  - Complements the relational data model concepts
    - Helps to visualize structure and content of data groups
      - entity is mapped to a relational table
    - Tool for conceptual data modeling (higher level representation)
  - Represented in an Entity Relationship Diagram (ERD)
    - Formalizes a way to describe relationships between groups of data

DBMS - Data models

- Define how data is connected to each other ; how they are processed and stored inside the system.
- Define how the logical structure of a database is modeled

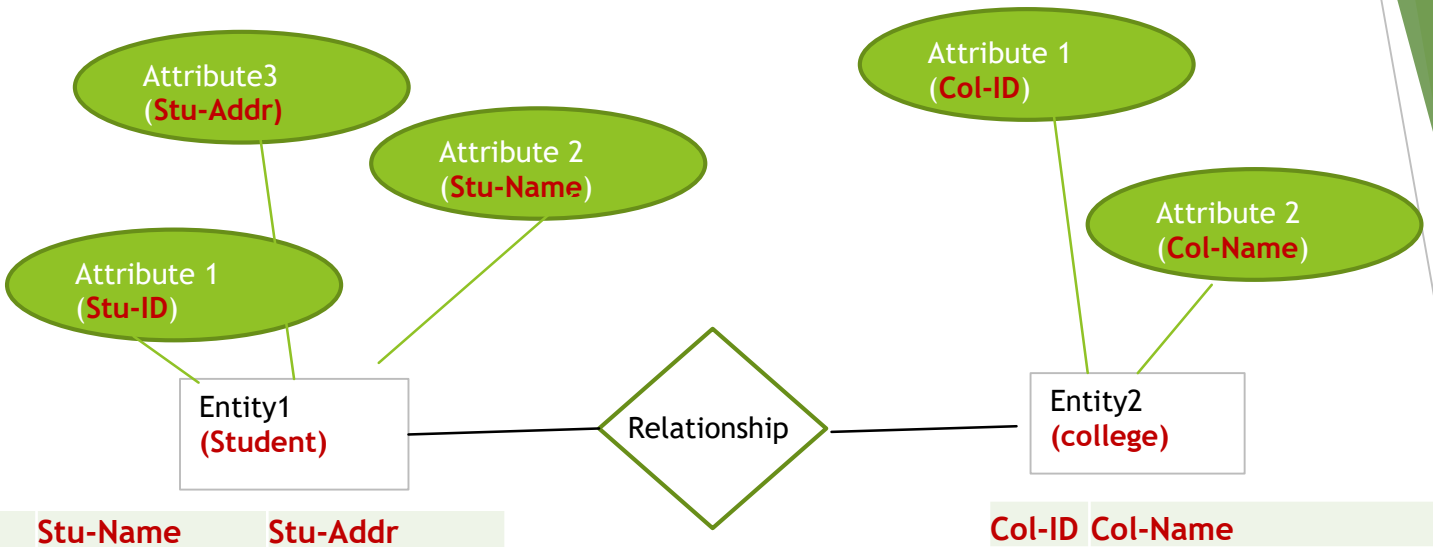
Entity-Relationship Model

- Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them.
- Best used for the conceptual design of a database
- Model is based on −
  - **Entities** and their *attributes*.
  - **Relationships** among entities



Sample E-R Diagram

DBMS - ER Model (Contd..)



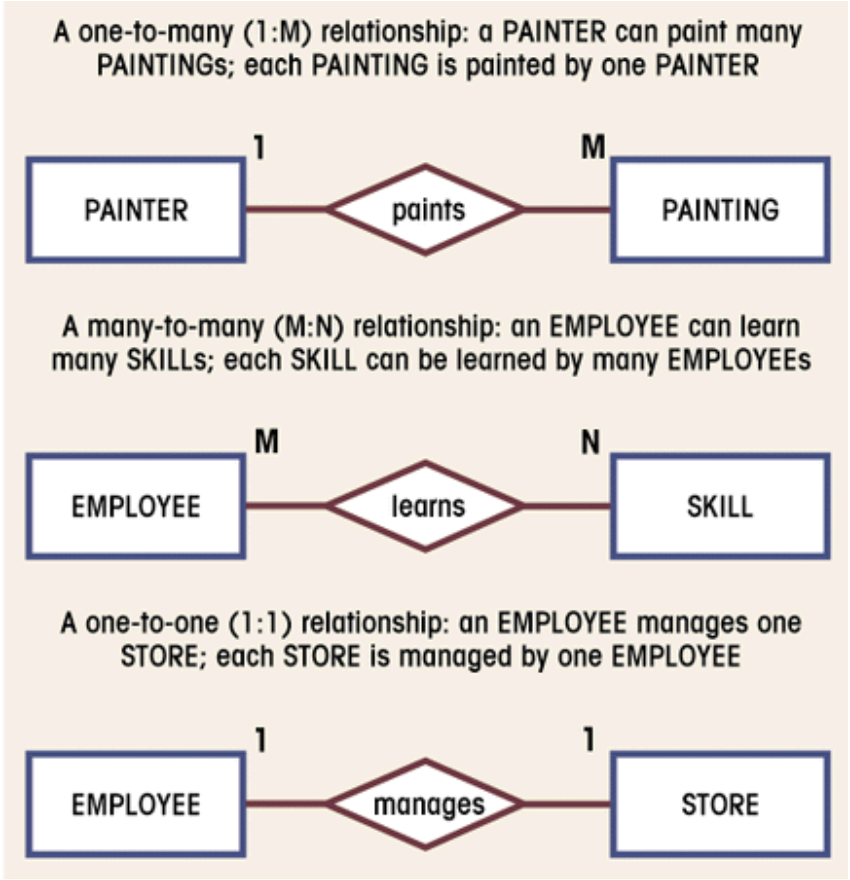
Stu-ID	Stu-Name	Stu-Addr
1001	John	Delhi
1002	Smitha	Kanpur
1003	Dinesh	Bengaluru
1004	Abraham	Chennai
1005	Krishna	Delhi
1006	Ankur	Hyderabad
1007	Deepa	Chennai
1008	Venkatesh	Assam

Domain

- One to one
- One to many
- Many to one
- Many to Many

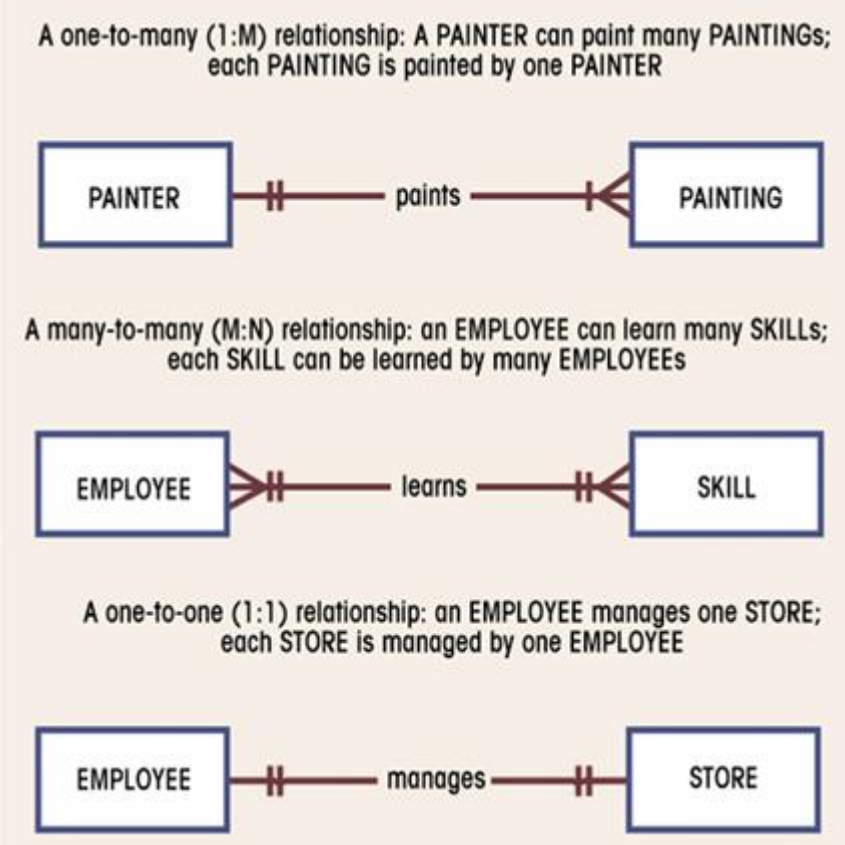
Col-ID	Col-Name
1	Govt.college
2	Reva college
3	Srinidhi college
4	Dayananda
5	NPS
6	APS
7	BMS
8	Acharya paatashala

E-R Diagram: Chen Model



- Entity
  - represented by a rectangle with its name in capital letters.
- Relationships
  - represented by an active or passive verb inside the diamond that connects the related entities.
- Connectivity
  - i.e., types of relationship
  - written next to each entity box.

E-R Diagram: Crow's Foot Model



- Entity
  - represented by a rectangle with its name in capital letters.
- Relationships
  - represented by an active or passive verb that connects the related entities.
- Connectivities
  - indicated by symbols next to entities.
    - 2 vertical lines for 1
    - "crow's foot" for M

E-R Model: Pros & Cons

- Advantages
  - Exceptional conceptual simplicity
    - easily viewed and understood representation of database
    - facilitates database design and management
  - Integration with the relational database model
    - enables better database design via conceptual modeling
- Disadvantages
  - Incomplete model on its own
    - Limited representational power
      - cannot model data constraints not tied to entity relationships
        - e.g. attribute constraints
      - cannot represent relationships between attributes within entities
    - No data manipulation language (e.g. SQL)
  - Loss of information content
    - Hard to include attributes in ERD

DBMS – Data Model – ER model

Relationship

- Defines an association among entities.
  - For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

Relationship Set

- A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

Degree of Relationship

- The number of participating entities in a relationship defines the degree of the relationship.
- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

Mapping Cardinalities

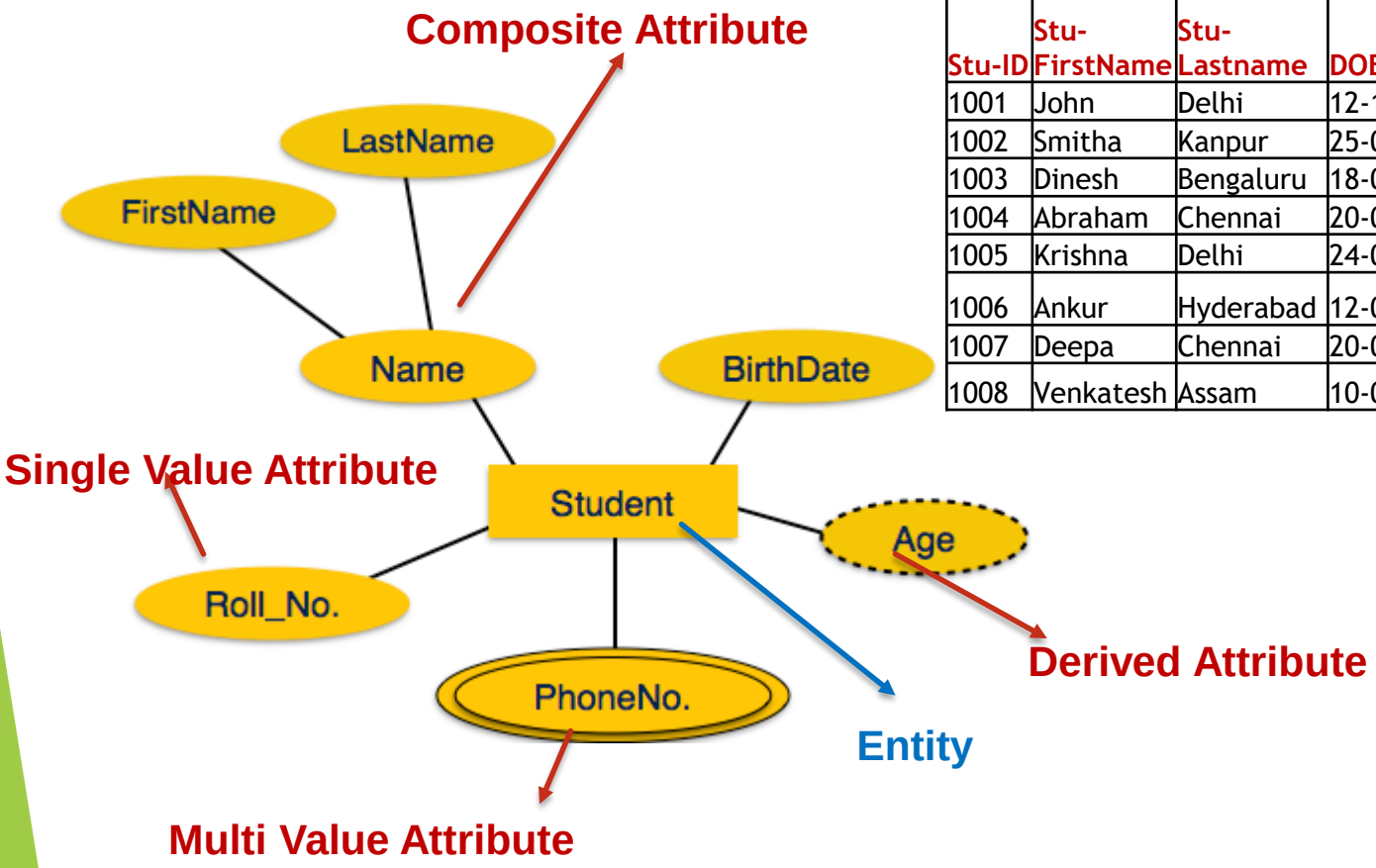
- **Cardinality** defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.
  - **One-to-one** – One entity from entity set A can be associated with at most one entity of entity set B and vice versa.
  - **One-to-many** – One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.
  - **Many-to-one** – More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A
  - **Many-to-many** – One entity from A can be associated with more than one entity from B and vice versa.

DBMS – Data Model – ER model

Types of Attributes

- **Simple attribute** – atomic values, which cannot be divided further.
  - For example, a student's phone number is an atomic value of 10 digits.
- **Composite attribute** – made of more than one simple attribute.
  - For example, a student's complete name may have first_name and last_name.
- **Derived attribute** – attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
  - For example, average_salary in a department; age can be derived from data_of_birth.
- **Single-value attribute** – contains single value.
  - For example – Social_Security_Number.
- **Multi-value attribute** – contain more than one values.
  - For example, a person can have more than one phone number, email_address, etc.

DBMS – Data Model – ER model



Stu-ID	Stu-FirstName	Stu-Lastname	DOB	Phone
1001	John	Delhi	12-10-1991	9865980011
1002	Smitha	Kanpur	25-03-1992	9665978001
1003	Dinesh	Bengaluru	18-06-1991	9865987603
1004	Abraham	Chennai	20-06-1992	9865987901
1005	Krishna	Delhi	24-09-1991	9868760011
1006	Ankur	Hyderabad	12-08-1992	9865786711
1007	Deepa	Chennai	20-07-1991	9865988979
1008	Venkatesh	Assam	10-09-1991	9865967811

DBMS – Data Model – ER model

- **Key** is an attribute or collection of attributes that uniquely identifies an entity among entity set.
  - For example, the roll_number of a student
- **Types of Keys**
  - **Super Key** – A set of attributes (one or more) that collectively identifies an entity in an entity set.
  - **Candidate Key** – A minimal super key. An entity set may have more than one candidate key.
  - **Primary Key** – one of the candidate keys chosen to uniquely identify the entity set.

Stu-ID	Stu-Name	Stu-Addr
1001	John	Delhi
1002	Smitha	Kanpur
1003	Dinesh	Bengaluru
1004	Abraham	Chennai
1005	Krishna	Delhi
1006	Ankur	Hyderabad
1007	Deepa	Chennai
1008	Venkatesh	Assam

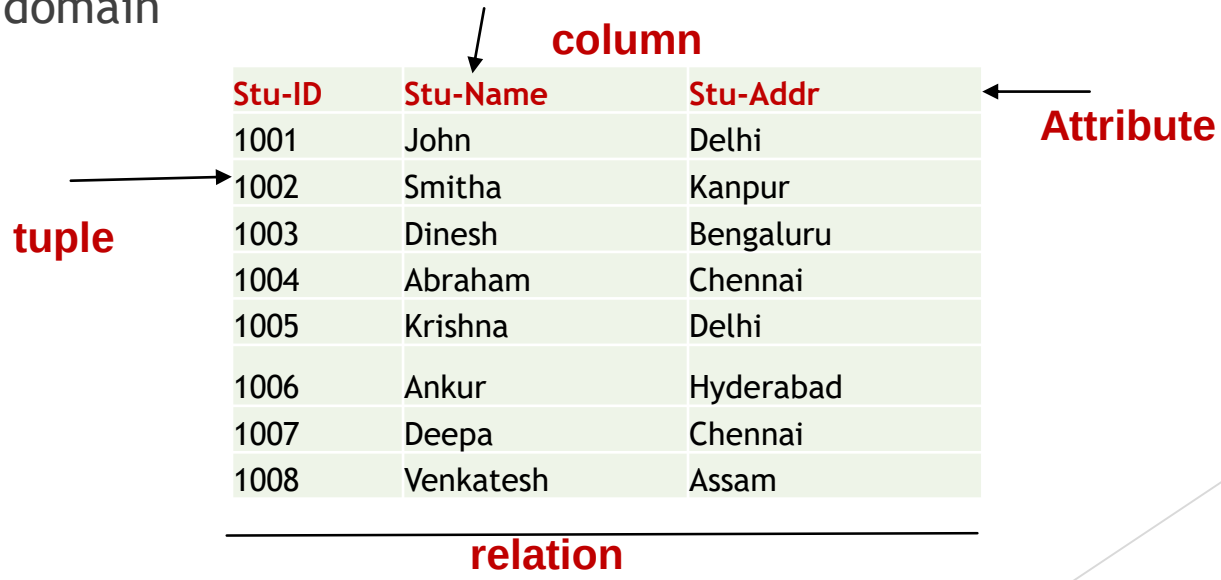
NoSQL

- Pros
  - open source (Cassandra, CouchDB, Hbase, MongoDB, Redis)
  - Elastic scaling
  - Key-value pairs, easy to use
  - Useful for statistical and real-time analysis of growing lists of elements (tweets, posts, comments)
- Cons
  - Security (No ACID: ACID (Atomicity, Consistency, Isolation, Durability)
  - No indexing support
  - Immature
  - Absence of standardization

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumber": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

DBMS - Data Model : Relational Model

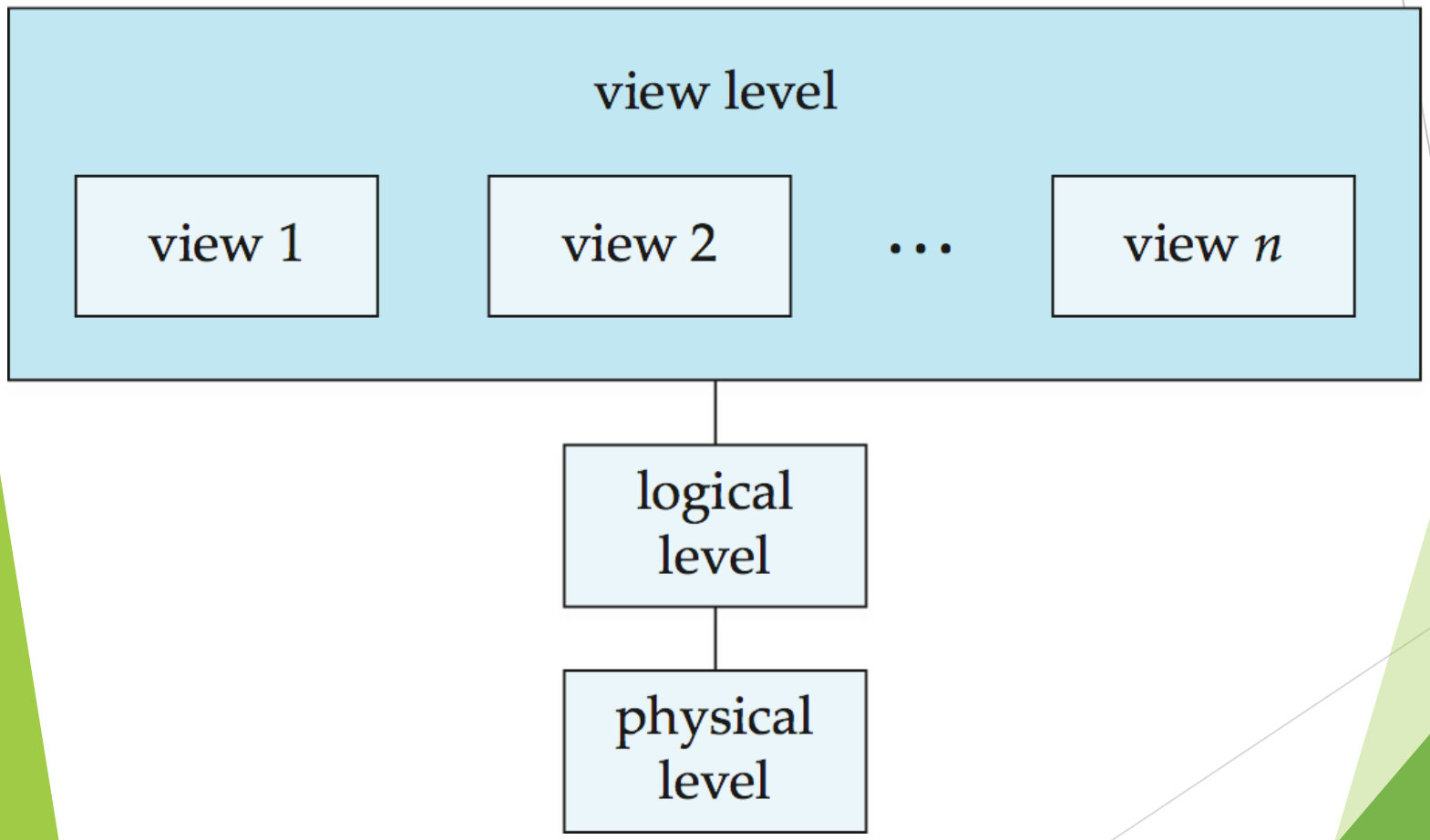
- Data is stored in tables called **relations**
- Table is defined in n-ary
- Each row in a relation contains a unique value
- Relations can be normalized
- In normalized relations, values saved are atomic values
- Each column in a relation contains values from a same domain



Stu-ID	Stu-Name	Stu-Addr
1001	John	Delhi
1002	Smitha	Kanpur
1003	Dinesh	Bengaluru
1004	Abraham	Chennai
1005	Krishna	Delhi
1006	Ankur	Hyderabad
1007	Deepa	Chennai
1008	Venkatesh	Assam

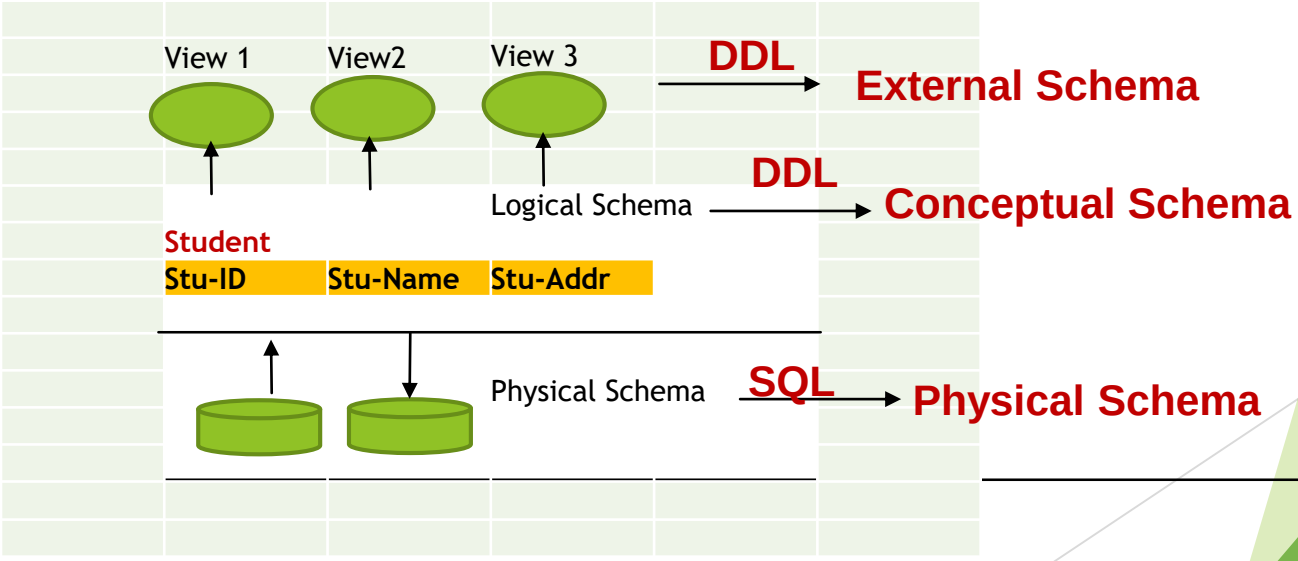
DBMS - Views

An architecture for a database system



DBMS - Data Schemas

- Represents the logical view of the entire database
- Designed before physical database is created
- Defines its entities and the relationship among them
- Defines how the data is organized
- Defines how the relations among them are associated.
- Formulates all the constraints that are to be applied on the data



DBMS - Levels of Abstraction

- **Physical Schema:** describes how a record is stored.

- **Logical Schema:** describes data stored in database(table), and the relationships among the data, constraints applied to data.

Students(Stu-ID : string;
Stu-name : string;
Stu-addr : string)

Stu-ID	Stu-Name	Stu-Addr
1001	John	Delhi
1002	Smitha	Kanpur
1003	Dinesh	Bengaluru
1004	Abraham	Chennai

- **View level:** application programs hide details of data types and information.

DBMS - Database Instances

- State of operational database with data at any given time.
- Snapshot of the database.
- Change with time
- Exists in a valid state (validations, constraints, and conditions that the database designers have imposed
  - Analogous to the value of a variable

DBMS -Data independence

- **Logical Data Independence**
  - mechanism, which liberalizes itself from actual data stored on the disk.
  - Any changes done to table format table format will not affect data residing on the disk.
- **Physical Data Independence**
  - the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Note: Similar to types and variables in programming languages

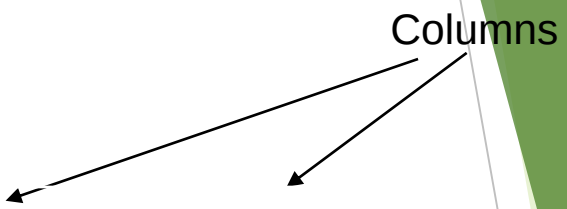
- All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.
- For example, in case we want to change or upgrade the storage system itself – suppose we want to replace hard-disks with SSD – it should not have any impact on the logical data or schemas.

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

- Data is stored in various tables.
- Tabular data in the relational model



Rows

Student ID	Student Name	Department Name	Course Name
1001	Arun	Mechanical	Engg.Maths
1002	Bhaskar	Electrical	Chemistry
1003	John	Computer Science	Cloud computing
1004	Kumar	Electronics	Data Analytics
1005	Sara	Electrical	DSP
1006	Abraham	Mechanical	CAD
1007	Anil Kumar	Computer Science	Programming Lang
1008	Sweta	Mechanical	Therrmo dynamics
1009	Rubeena	Computer Science	AI
1010	Neela	Civil	Structures

Students Database

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

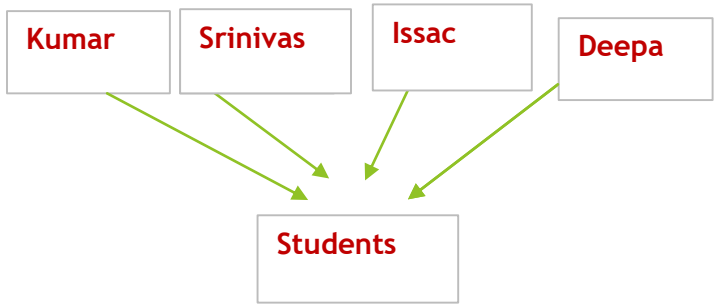
dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

DBMS - Generalization

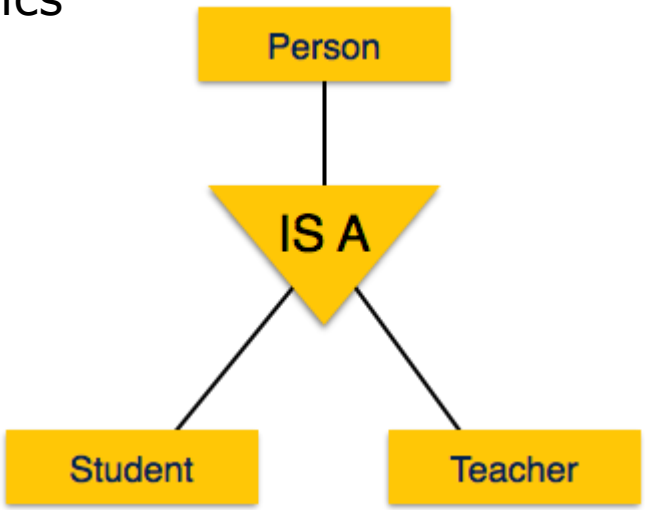
Generalization:

- entities are clubbed together to represent a more generalized view
- generalized entities contain the properties of all the generalized entities



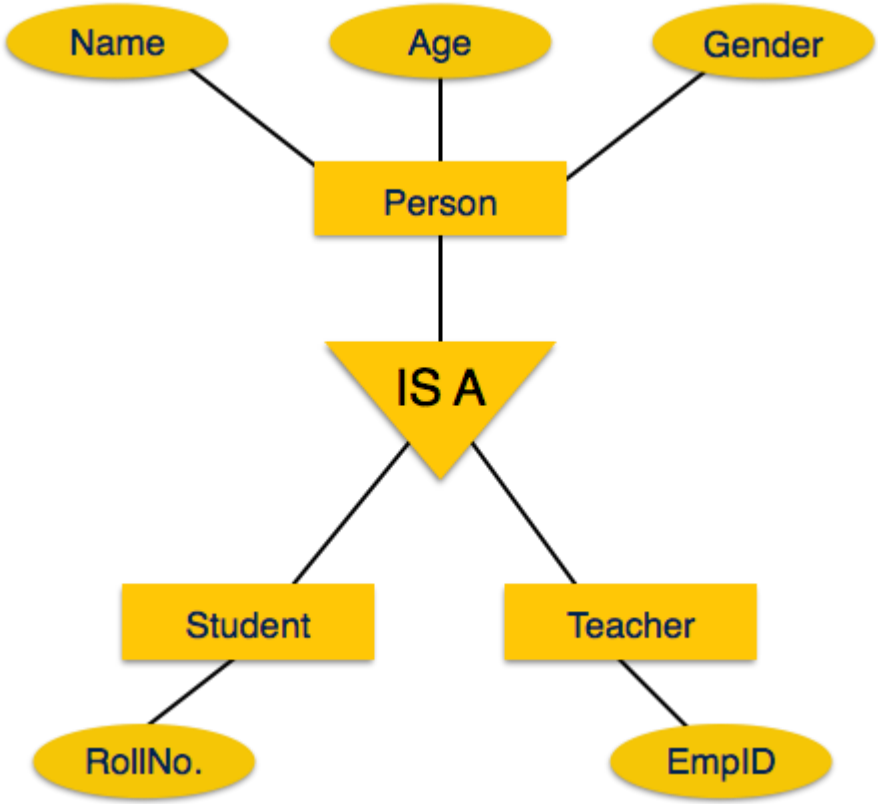
Specialization

- Specialization is the opposite of generalization.
- Group of entities is divided into sub-groups based on their characteristics



DBMS - Inheritance

- Important feature of Generalization and Specialization.
- Allows lower-level entities to inherit the attributes of higher-level entities



Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design (Cont.)

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Exercise No 1 – ER diagrams

1. Airline Booking System
   - Identify 5 Entities and 5 relationships
   - Assume 10 Domain values
   - Draw ER diagram for Entities and Relationships

2. Hospital Patients management system
   - Identify 5 Entities and 5 relationships
   - Assume 10 Domain values
   - Draw ER diagram for Entities and Relationships