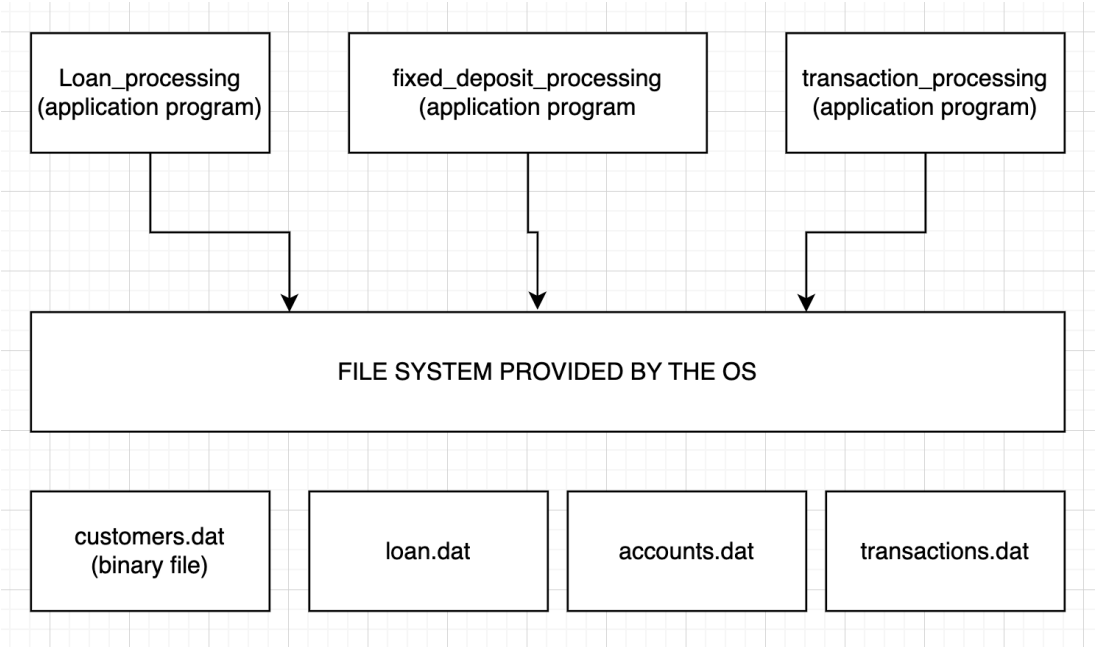
RDBMS Concepts

DB and DBMS

- Data
    - Meaningful facts, text, graphics, audio, video, and more complex values
    - A piece of information
- Database
    - An organized collection of logically related data
    - A collection of entities
    - Represents some aspect of REAL world
        - AKA miniworld
        - Example:
            - HR MANAGEMENT
            - ORDER PROCESSING SYSTEM
            - PAYROLL SYSTEM
            - ECOMMERCE APP
    - Some amount of interaction with events in the real world
    - Changes to the miniworld are reflected in the database
- Information
    - Processed data
    - Users are interested in information and not data
    - Applications use/access the data from some storage media, and provide very useful information
- Metadata
    - Data that describes other data
    - Example:
        - data → employee_id = 1234
        - metadata → type -> INTEGER, min -> 0, max-> 99999

Traditional (legacy) method of data storage

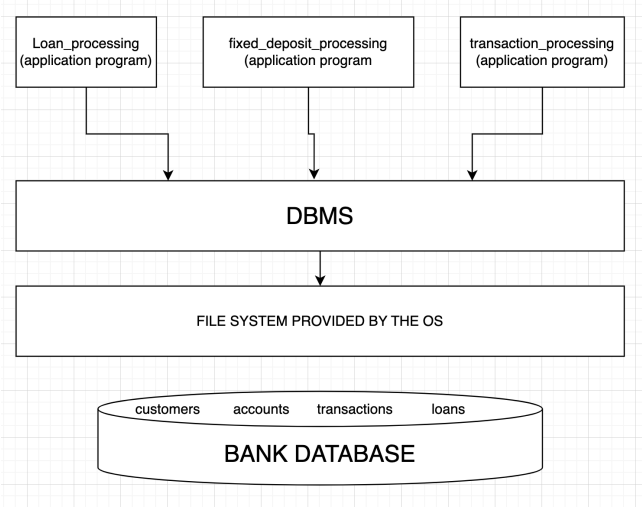


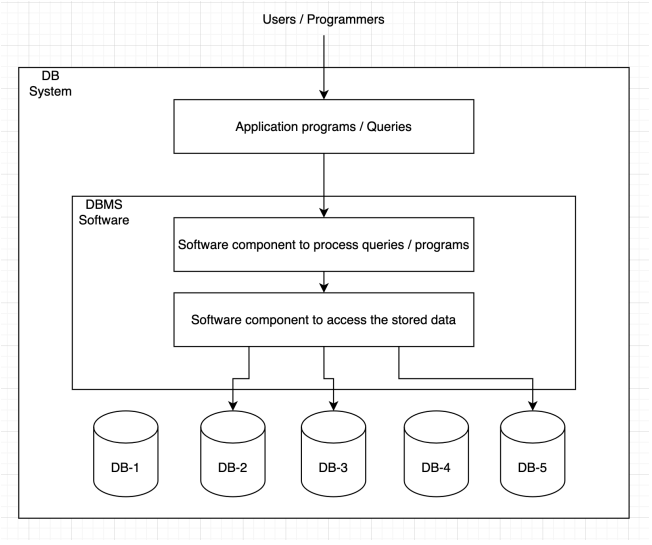
Limitations / downside of this approach:

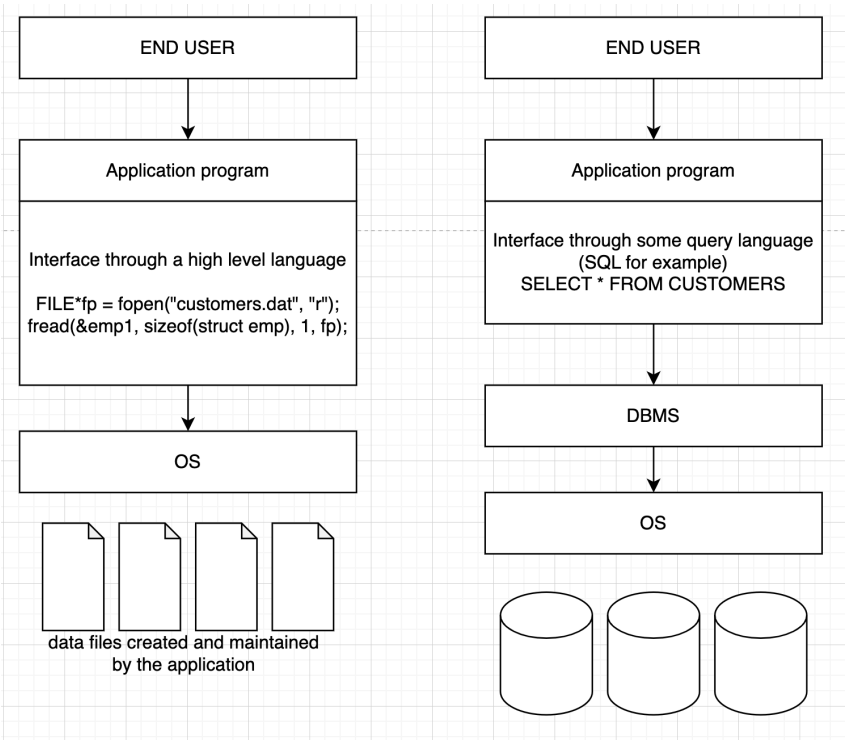
- Data dependence
- Data redundancy (duplication of data)
- In a networked environment, limited data sharing ability
- Development time is lengthy
- Excessive program maintenance

What is the alternative to this? - DBMS

- Database Management System
- Controlling redundancy
- Sharing of data with other applications
- Restrict unauthorized access
- Provide multiple interfaces
- Enforcing of constraints
- Provide facility for backup and recovery of data
- Enforce standards
- Reduce application development time
- Data should be available all the time, and must be up-to-date
- Provide a common language for communication (across different implementations of DBMS)

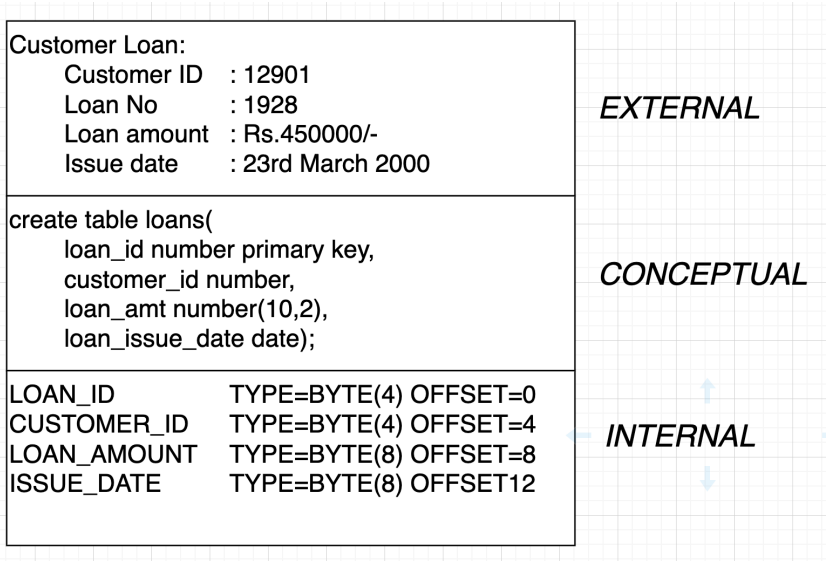






Different views of a data

1. External view (individual users)
2. Conceptual view
3. Internal / Storage view



Data model

- A Collection of Concepts
- Describe the structure of the database
- Describe the way “People” perceive (picturise) data

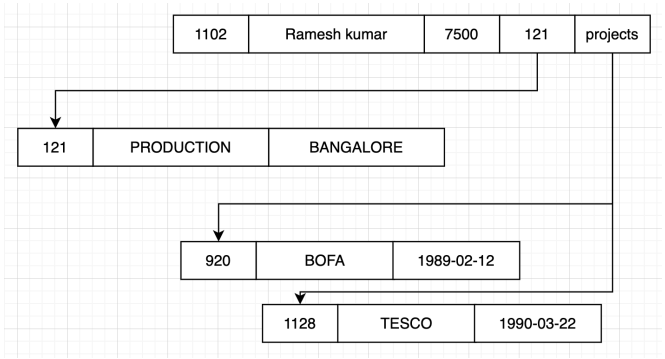
Different data model implementations:

1. Hierarchical (record based) data model (legacy)
2. Network data model (legacy)
3. Relational data model

```
struct employee {
       int id;
       char name[50];
       float salary;
       struct department dept;
       struct project* projects;
};

struct department {
       int id;
       char name[50];
       char location[50];
};

struct project {
       int id;
       char name[50];
       date start_date;
};
```



Few drawbacks of hierarchical model

- Knowledge of the structure and its physical representation is required by the developer
- Does not support for modifying the structure
- Time consuming - processing is done one record at a time

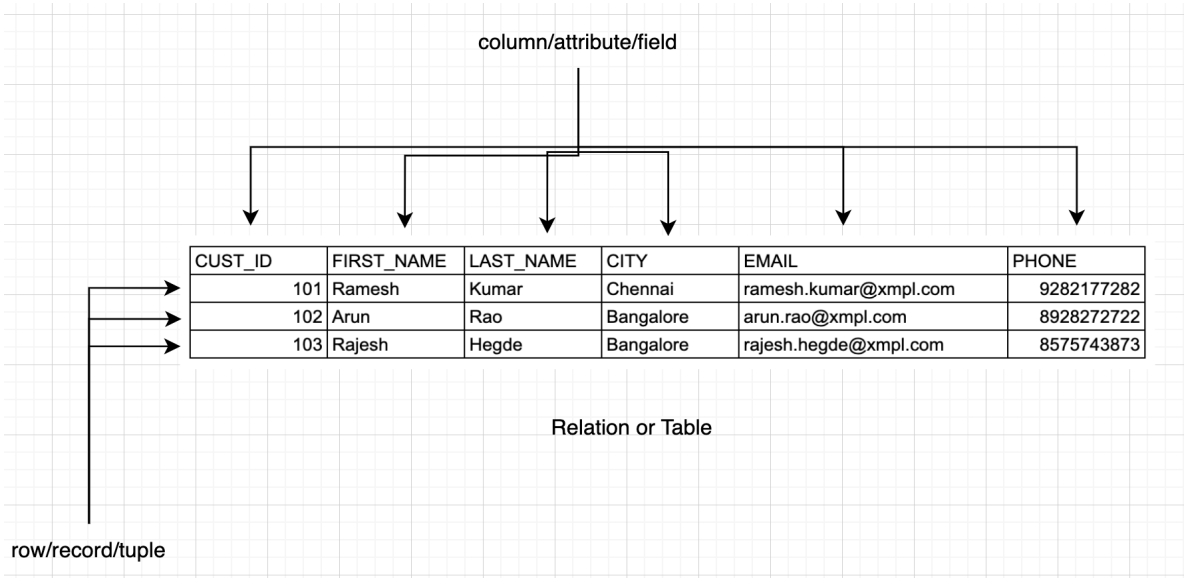
Network model (Self study)

Record based Relational data model

Terminologies:

1. Row (record or tuple)
2. Column (attribute or field)
3. Table (Relation)

External view of a Relation / Table:



Some key pointers:

- A database ia collection of related tables (of the miniworld)
    - For example, in a HR management system, the relations (tables):
        - EMPLOYEES
        - DEPARTMENTS
        - HOLIDAYS
        - EMPLOYEE_LEAVES
        - JOBS
    - In a order processing system:
        - PRODUCTS
        - SUPPLIERS

        - CUSTOMERS
        - CUSTOMER_ORDERS
- Every relation (table) has a unique name in a database
- Every attribute in a relation has a unique name
- The value of an attribute is atomic
- Every row in a relation is also unique
- Sometimes a value of an attribute may be unknown or have no value - this is represented using a special value called NULL
    - It is neither zero nor blank nor empty string
- Keys
    - Candidate key - set of one or more columns that may uniquely identify a row in a given table
        - email_id, phone, emp_no, passport_no, aadhar_no - all of these may uniquely identify an employee in a table full of employees
    - Primary key - Designated column among the candidate keys. The value of this may never be NULL nor duplicate
    - Composite primary key - When a primary key is a combination of more than one column
    - Foreign key - It is a regular column in one table but the same column exists in another table as a primary key. For example, DEPT_NO is the primary key in the DEPARTMENTS table, but may exist in the EMPLOYEES table (indicating the department number of the employee) as a FOREIGN KEY
- Integrity Constraints
    - Domain (field level) integrity
        - Allowable values for an attribute
        - Data type, NULL or NOT NULL, CHECK
    - Entity Integrity
        - No primary key value may be null (unique as well)
    - Referential Integrity
        - When FOREIGN KEY is applied on a column
        - Cascade UPDATE or cascade DELETE rules

ER (Entity Relationship) Model

Constitutes:

- Entity
    - Source of information
    - Person, Customer, Employee, Product, …
    - Nouns
- Attribute
    - Data of an entity
    - For the entity Customer
        - name, address, city, state, country, email, phone, …

- Relationships
    - Relationship between two or more entities
    - CUSTOMER ***places order*** for PRODUCTS
    - STUDENT ***registers*** for a COURSE
    - EMPLOYEE ***applies*** for a LEAVE
    - DOCTOR ***prescribes*** MEDICINE for a PATIENT

Simple Vs Composite attributes

- Simple attribute
    - Can not be divided into more simpler attributes
    - For example, *age* or *first_name* of a person
- Composite attribute
    - Can be split into more attributes
    - For example, *address* of a CUSTOMER can be split into HOUSE_NAME_NO, STREET, CITY, STATE, COUNTRY, PINCODE
    - *date_of_birth* of an EMPLOYEE can be split into DAY, MONTH, YEAR

Single valued Vs Multi-Valued attributes

- Single valued:
    - Can take a single value for each entity instance
    - For example, for an employee, the *salary* attribute may take only one value
- Multi-valued:
    - Can take multiple values (usually comma separated values)
    - For example, the *skill_set* attribute for an EMPLOYEE can be
        - “java, c, c++, c#”
        - “python, php, mysql, html”
        - “html, javascript, css, angular, react”

Stored Vs Derived attribute

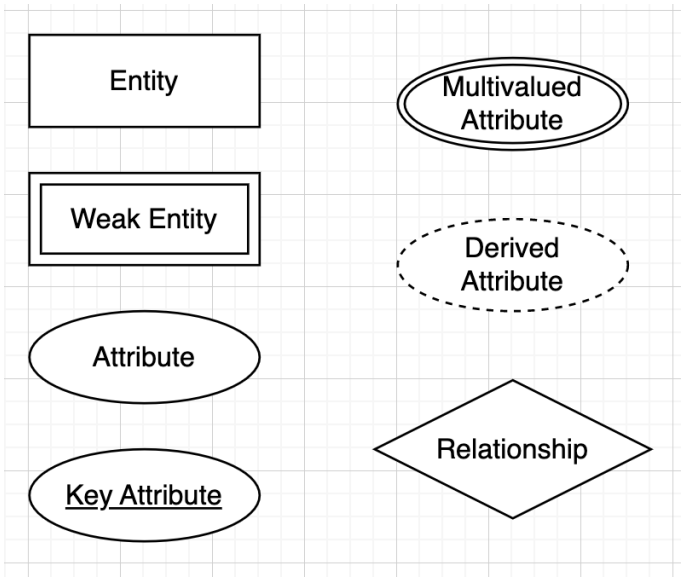
- Stored attributes:
    - Usually a fixed value, stored in the table permanently
    - For example, *first_name* of a CUSTOMER
- Derived attributes:
    - Calculated periodically and stored, based on a value of another attribute
    - For example, *age* of a PERSON is calculated based on the *date_of_birth*
    - *order_total* for an ORDER placed by a CUSTOMER is a calculated field (based on the products, quantities, discounts, etc.

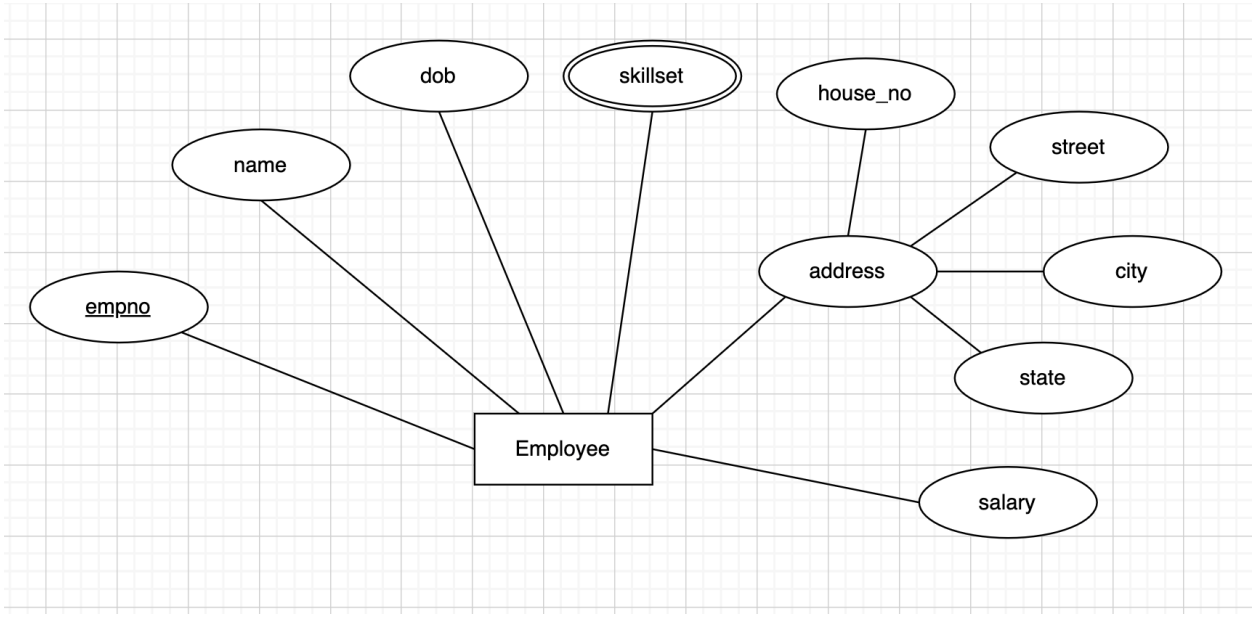
Regular Vs Weak entities

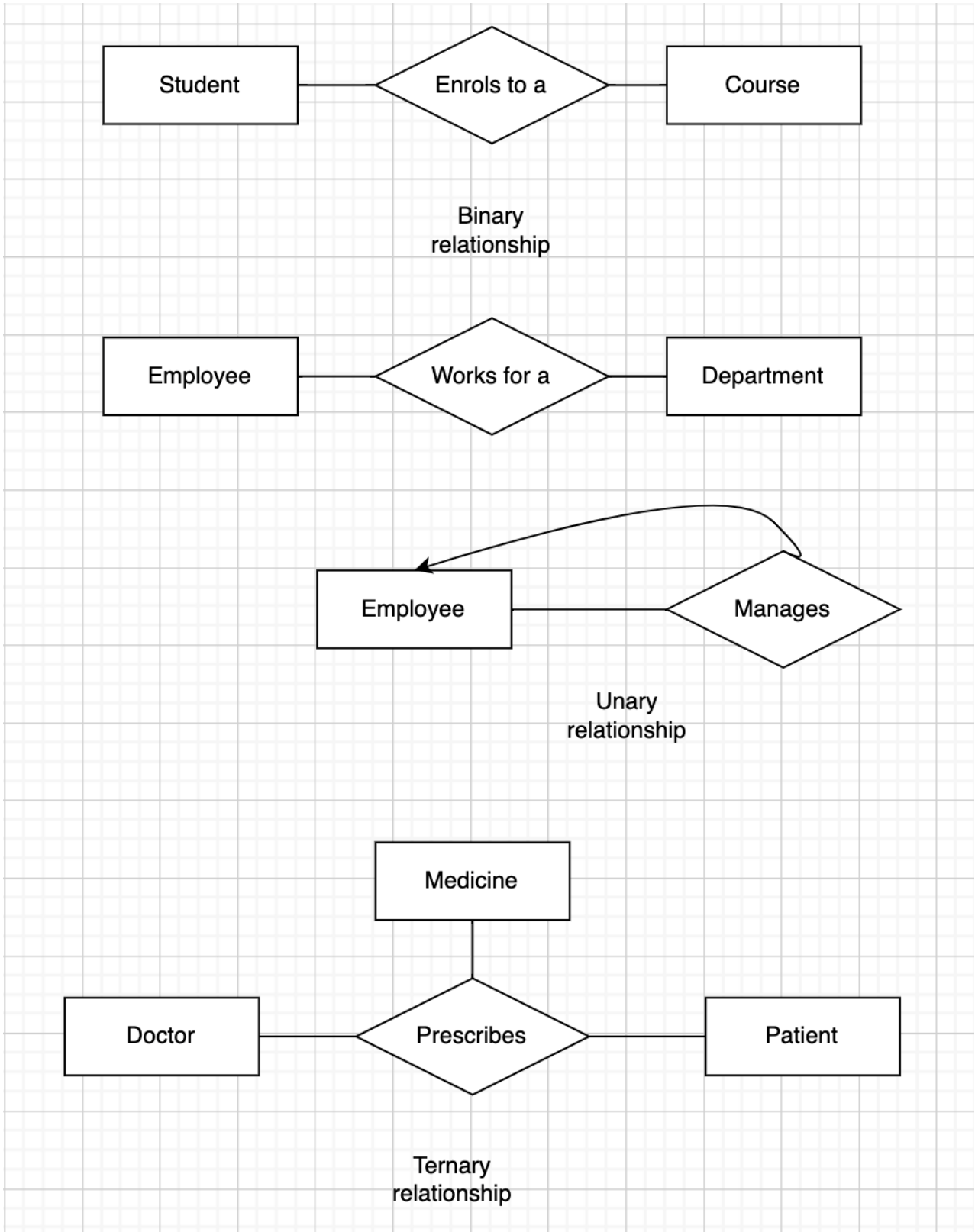
- Regular entity
    - Entity that has its own PRIMARY KEY attribute
    - For example, EMPLOYEE, PRODUCT, CUSTOMER, etc
- Weak entity:
    - An entity that does not have own PRIMARY KEY, but uses the PRIMARY KEY of another entity
    - For example, *NOMINEE* of an insurance policy holder
    - Very rarely found

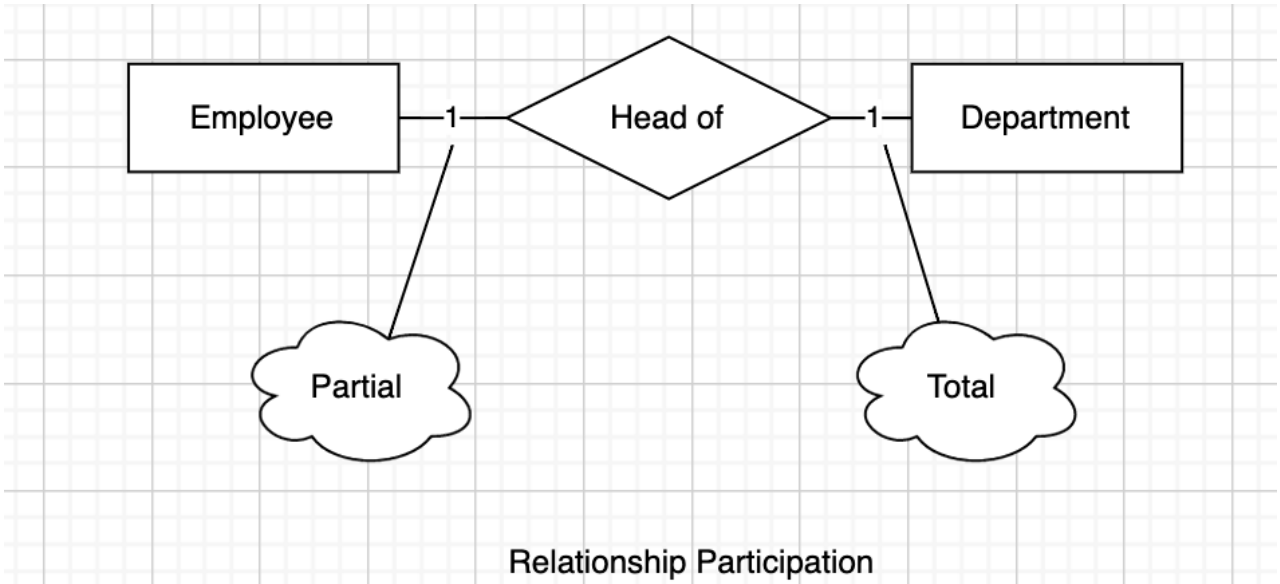
Relationships

- A relationship between two entity types defines the set of all associations between those two entities
- An instance of the relationship between these two entities, is called *a relationship instance*
- Degree:
    - Unary → Only one entity is participating in the relationship
        - *Employee → Manager_of_employee*
    - Binary → Two entities are participating in the relationship
        - *Employee → works_in → Department*
    - Ternary → Three entities are participating in the relationship
        - *Doctor → prescribes → Medicine for Patient*
        - *Customer → purchases → Product* delivered by *Courier*
- Cardinality:
    - One-To-One
        - EMPLOYEE → LAPTOP
        - HOD → DEPARTMENT
    - One-To-Many
        - DEPARTMENT → EMPLOYEE
    - Many-To-One
        - EMPLOYEE → DEPARTMENT
    - Many-To-Many
        - EMPLOYEE → PROJECT
        - one-to-many from both sides
- Relationship participation
    - Total
        - Every entity instance must be connected through the relationship to another instance of the participating entity types
        - Every employees *belong to* department/s
    - Partial
        - All entity instances need not participate in the relationship
        - Employee *heads a* department
            - Department → total participation (all departments have HOD)
            - Employee → partial participation (all employees cannot be HOD)

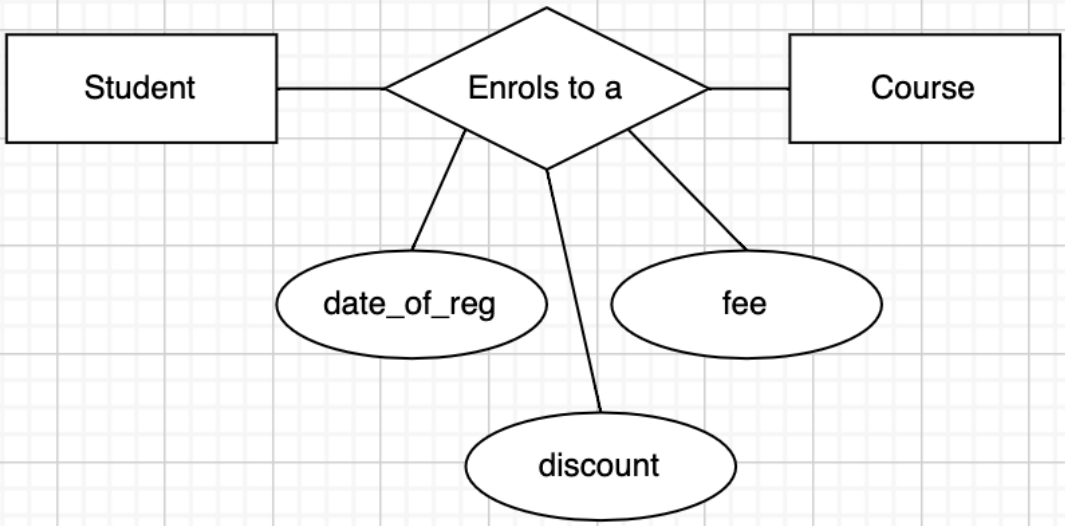

Entity
Multivalued Attribute
Weak Entity
Derived Attribute
Attribute
Relationship
Key Attribute


dob
skillset
house_no
name
street
address
city
empno
state
Employee
salary


Student
Enrols to a
Course
Binary relationship
Employee
Works for a
Department
Employee
Manages
Unary relationship
Medicine
Doctor
Prescribes
Patient
Ternary relationship



Relationship Participation



Attributes of a relationship