25CS305
Database Management Systems

M. NARMADHA,
Assistant Professor,
Computer Science and Engineering

NARSIMHA REDDY ENGINEERING COLLEGE
UGC AUTONOMOUS INSTITUTION
Maisammaguda (V), Kompally - 500100, Secunderabad, Telangana State, India
Accredited by NBA & NAAC with 'A' Grade
Approved by AICTE
Permanently affiliated to JNTUH

UNIT-I

What is data? A Historical perspective

A collection of facts from which conclusion may be drawn.

Data is often obtained as a result of recordings or observations.

- **Data is the plural form of datum.**
- **The temperature of the days is data.**

Temperature of the days

When this data is to be collected, a system or person monitors the daily temperatures and records it.

- Finally when it is to be converted into meaningful information, the patterns in the temperatures are analyzed and a conclusion about the temperature is arrived at.
- So information obtained is a result of analysis, communication, or investigation.

Properties of Data

Data should be well organized.

Data should be related.

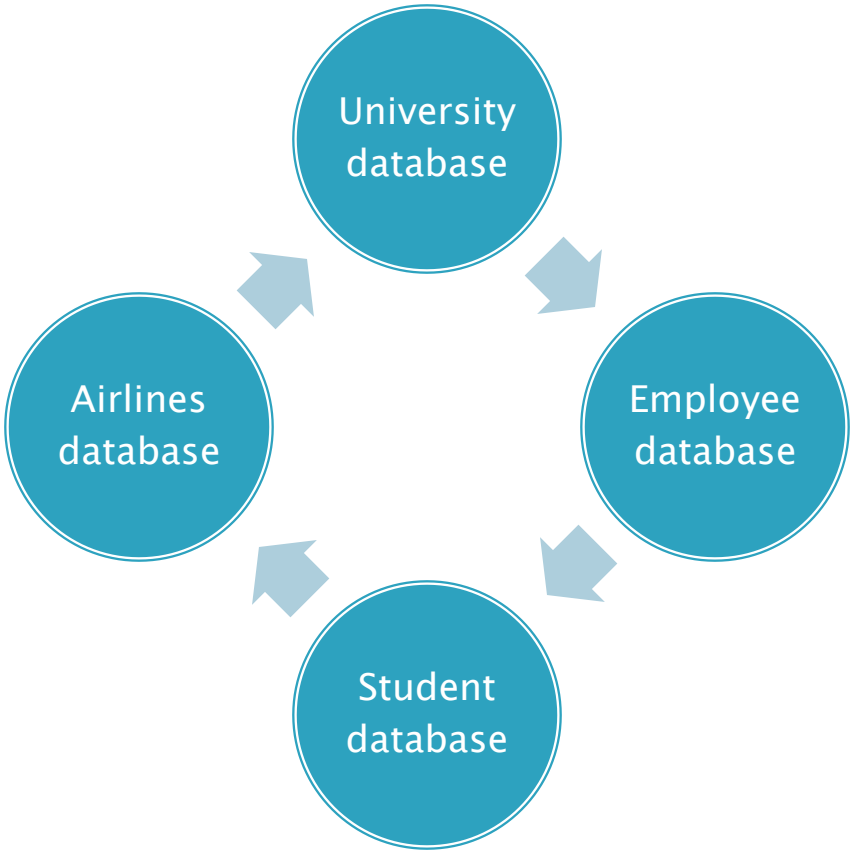
Data should be accessible in any order.

One data should be stored minimum number of times.

What is a Database?

•Database is a collection of related data, that contains information relevant to an enterprise.

For example:



PROPERTIES OF A DATABASE

A database represents some aspect of the real world, sometimes called the mini world or the universe of discourse (UoD).

A database is a logically coherent collection of data with some inherent meaning.

A database is designed, built and populated with data for a specific purpose.

What is Database Management System (DBMS)?

A database management system (DBMS) is a collection of programs that enables users to create & maintain a database. It facilitates the definition, creation and manipulation of the database.

Definition – it holds only structure of database, not the data. It involves specifying the data types, structures & constraints for the data to be stored in the database.

Creation –it is the inputting of actual data in the database. It involves storing the data itself on some storage medium that is controlled by the DBMS.

Manipulation-it includes functions such as updation, insertion, deletion, retrieval of specific data and generating reports from the data.

Typical DBMS Functionality

- Define a database : in terms of data types, structures and constraints
- Construct or Load the Database: on a secondary storage medium
- Manipulating the database : querying, generating reports, insertions, deletions and modifications to its content
- Concurrent Processing and Sharing by a set of users and programs: yet, keeping all data valid and consistent

Typical DBMS Functionality

Other features:

- Protection or Security measures to prevent unauthorized access
- “Active” processing to take internal actions on data Presentation and Visualization of data

Database System

The database and the DBMS together is called the database system.

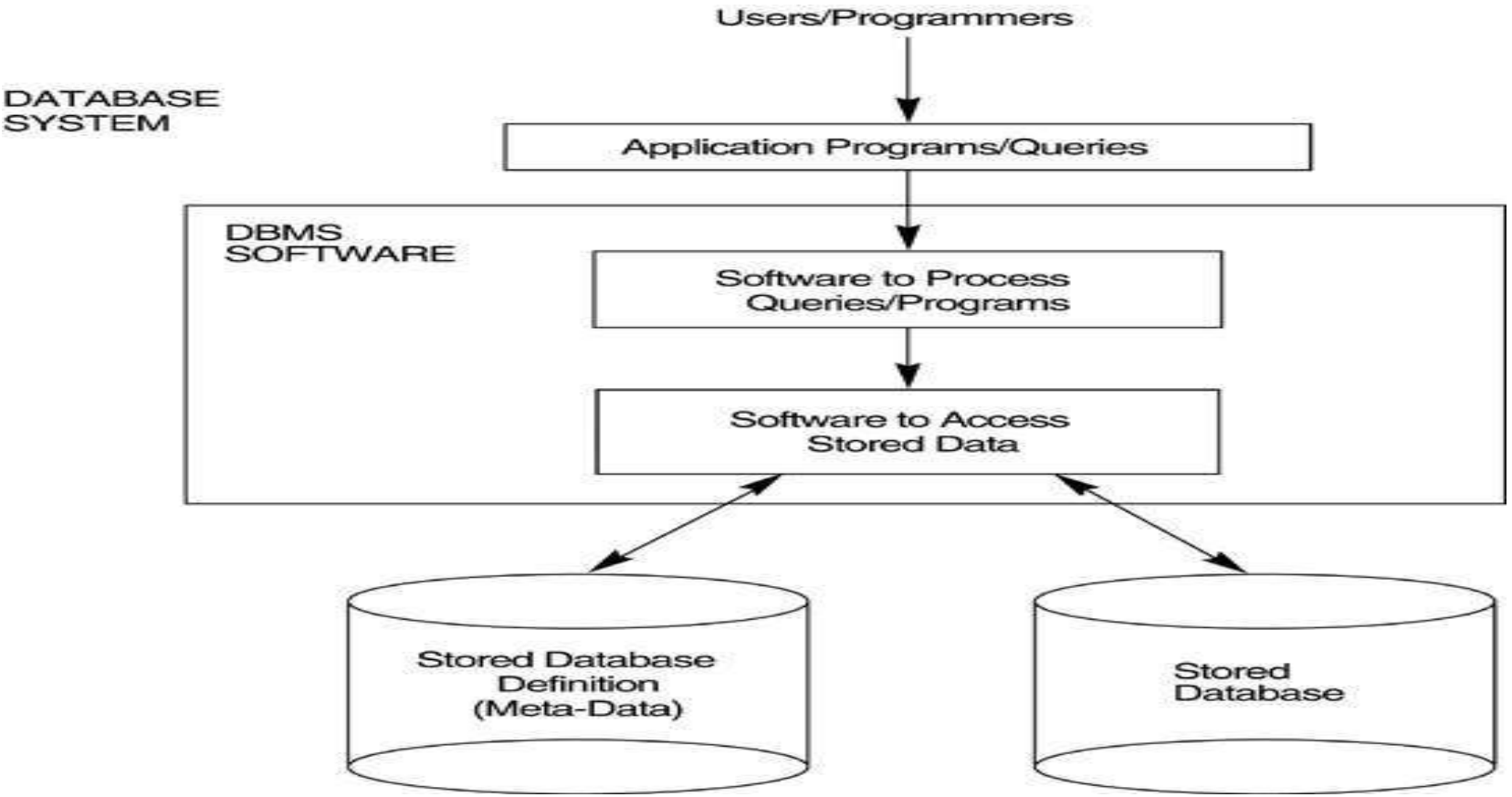
Database systems are designed to manage large bodies of information.

- It involves both defining structures for storage of information & providing mechanisms for the manipulation of information.

Database system must ensure the safety of the information stored.

Meta data- it is the data about the data. It contains the structure of the database as well as the physical location of the database.

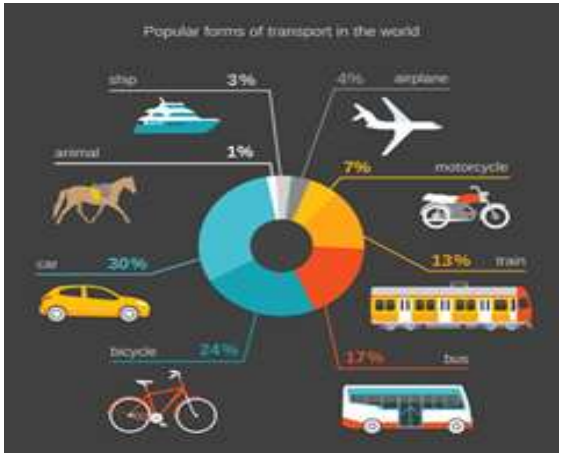
A simplified database system environment



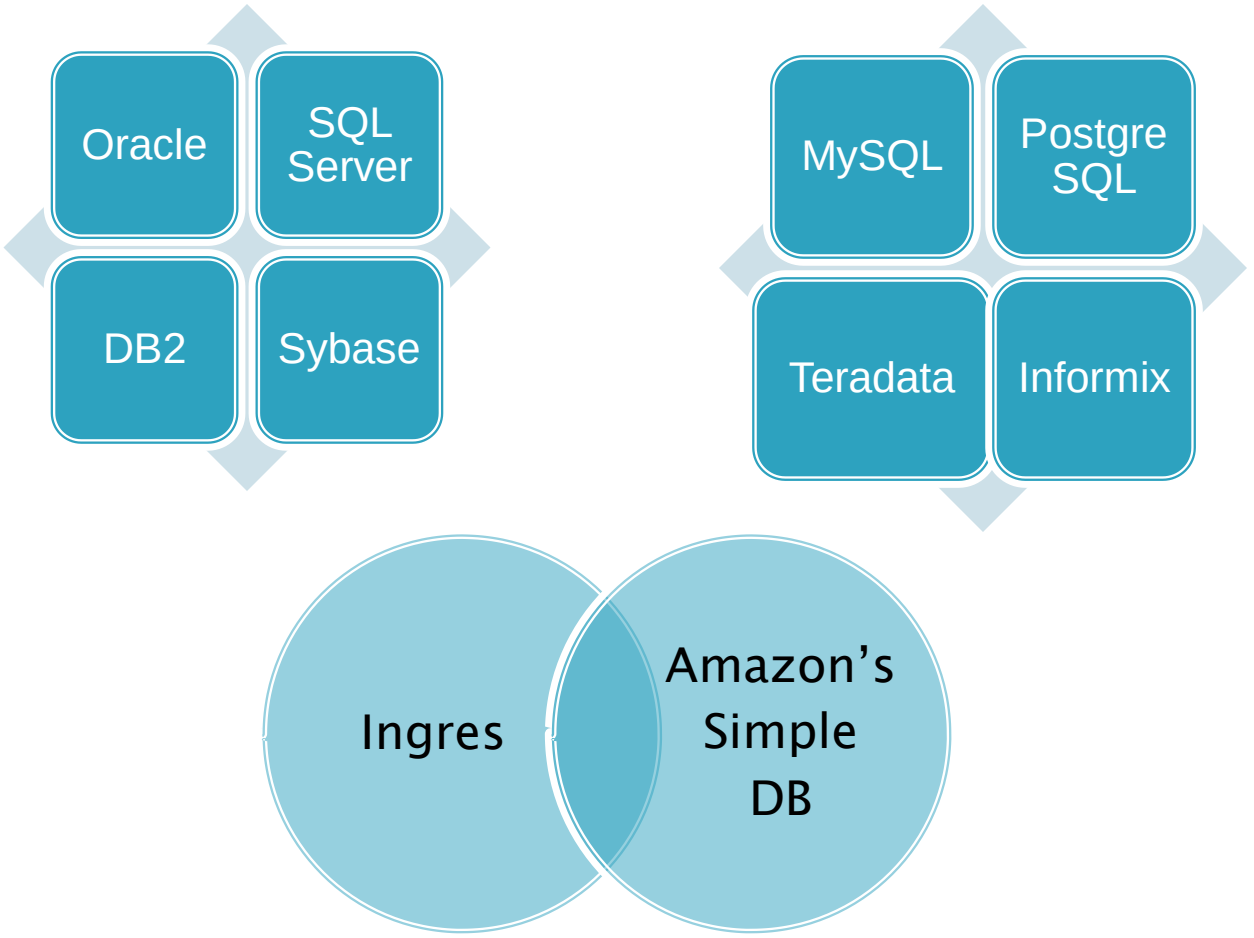
Database System Applications

- **Banking-** for customer information, accounts & loans, and banking transactions.
- **Airlines-**for reservations & schedule information.
- **Universities-**for student information, course registration and grades.
- **Credit card transactions-**for purchases on credit cards & generation of monthly statements.
- **Telecommunication-**for keeping records of calls made, generating monthly bills, maintaining balances, information about communication networks.
- **Finance-**for storing information about holdings, sales & purchases of financial instruments such as stocks & bonds.
- **Sales-**for customer, product and purchase information.
- **Manufacturing-**for management of supply chain & for tracking production of items in factories.
- **Human resources-**for information about employees, salaries, payroll taxes and benefits

Applications of DBMS



Database system



Traditional File systems

Before the evolution of DBMS, organizations used to store information in file systems.

- The system stores permanent records in various files & it need application program to extract records , or to add or delete records .

In traditional file processing, each user defines and implements the files needed for a specific application.

Traditional file system

For example, one user, the grade reporting office, may keep a file on students and their grades. Programs to print a student's transcript and to enter new grades into the file are implemented.

A second user, the accounting office, may keep track of students' fees and their payments.

- Although both users are interested in data about students, each user maintains separate files and programs to manipulate these files because each requires some data not available from the other user's files.

This redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common data up-to-date.

File System VS DBMS

Data redundancy and inconsistency

- Different →Programmers
- → Files
- →Structures
- →Programming Languages
- Redundancy
- Higher Storage and access cost

Difficulty in accessing the data

- Find all the students who gained 25 points in Hacker rank
- Find out all students who cleared all the subjects
- File systems are NOT convenient and efficient
- Cannot offer needed data-retrieval easily

Data isolation

- Different -> Programmers
- -> file
- -> Structures
- Files are stored at different locations
- So data isolation is difficult in file system

Integrity problems

- Consistency constraints
- Ex: student cant have less attendance
- Easy to maintain integrity in software but not in file system
- IN DBMS new constraints can be added whenever required

Atomicity problems

- Difficult to ensure atomicity in file-processing
- Restoration of data is difficult

Concurrent-access anomalies

- Concurrency is good
- May lead to inconsistency
- Example: Accessing a Google document were permission is given to everyone to read & write the document

- Data access
- Authentication and Authorization
- Finance personnel should not access Academic records
- Enforcing access constraints in file systems is difficult

Advantages of DBMS

- Controlling Redundancy
- Restricting Unauthorized Access
- Providing Storage Structures for Efficient Query Processing
- Permitting Inference and Actions using Rules
- Providing Multiple User Interfaces
- Representing Complex Relationship among Data
- Enforcing Integrity Constraints
- Providing Backup and Recovery

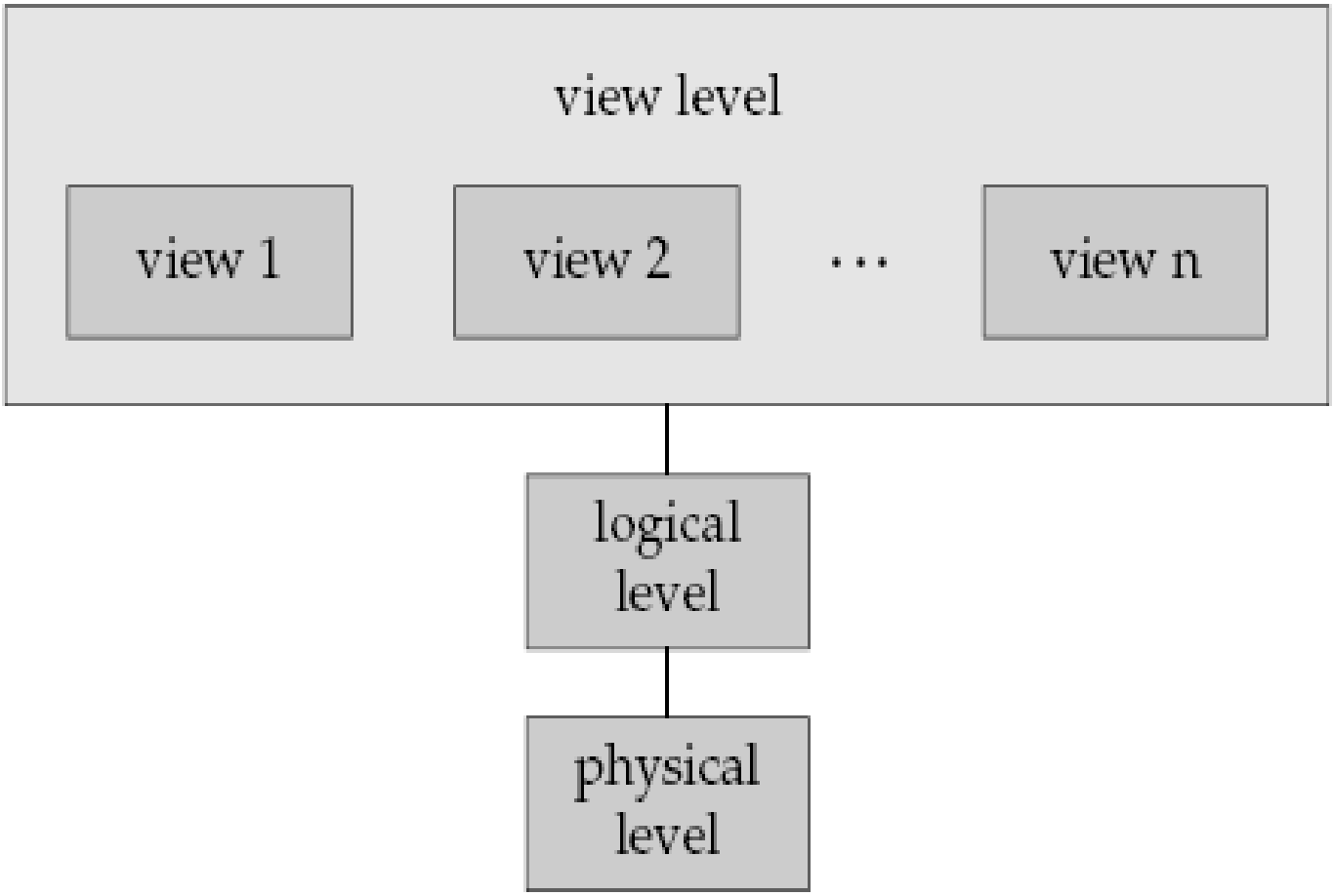
Data Abstraction

Data abstraction

Method of hiding the actual (complex) details from users is called as data abstraction.

That is, the DBMS system hides certain details of how the data are stored and maintained.

Levels of data abstraction



Physical level

Example:

Customer: with customer-id, customer-name, customer- street, customer-city

Logical level

It is the next level of abstraction & describes what data are stored in database & what relationship exists between various data.

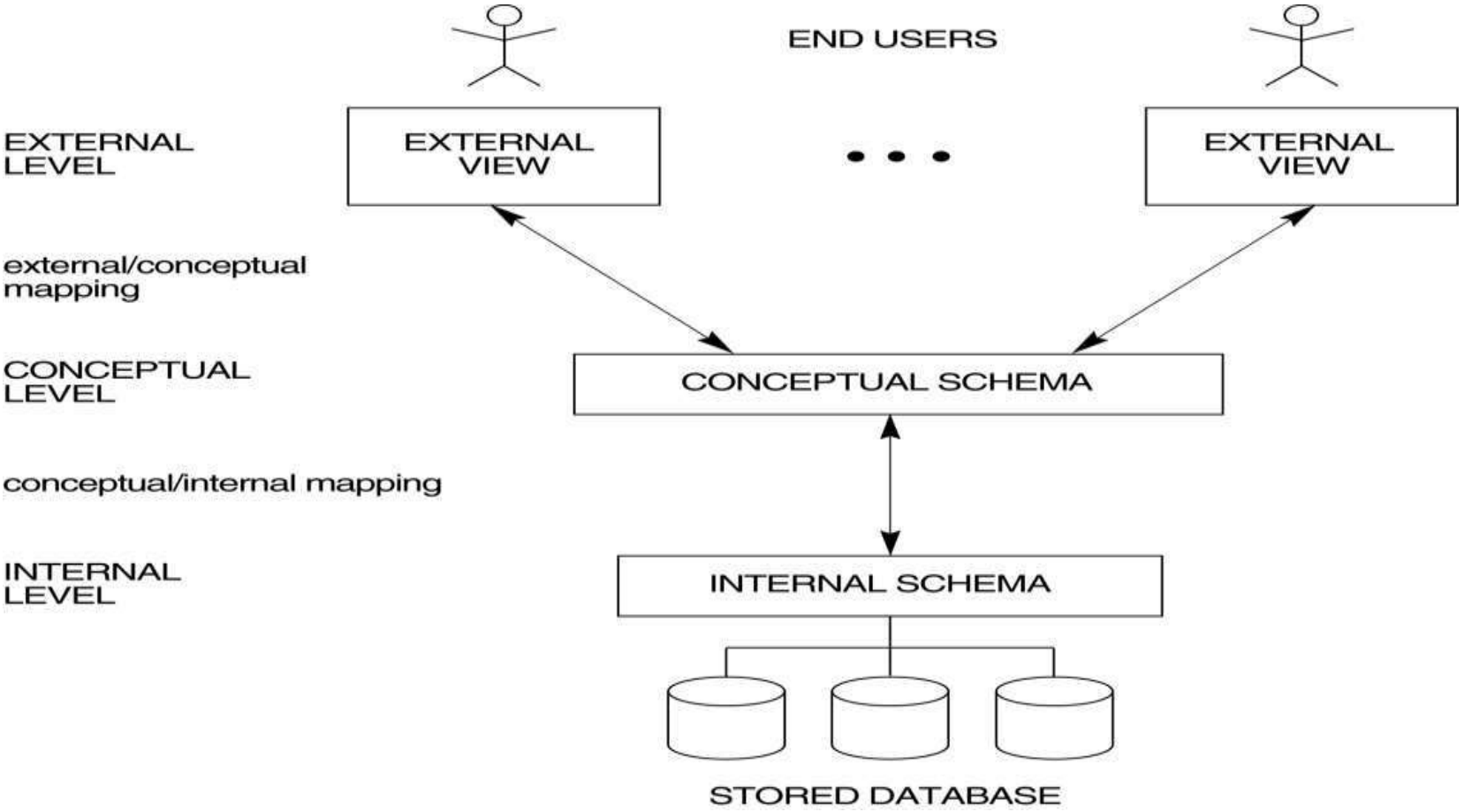
View level

This level contains the actual data which is shown to the users.

This is the highest level of abstraction & the user of this level need not know the actual details of data storage.

Structure of a DBMS

ANSI-SPARC 3-level DBMS Architecture



Three-schema architecture

here ANSI-SPARC stands for America National Standards Institute, Standards Planning And Requirements Committee.

The three-schema architecture is a convenient tool for the user to visualize the schema levels in a database system.

In this architecture, schemas can be defined at the following three levels:

- **Internal schema/Physical schema**
- **Conceptual schema**
- **External schema**

The internal level has an internal schema, which describes the physical storage structure of the database.

The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

The external or view level includes a number of external schemas or user views.

The processes of transforming requests and results between levels are called mappings.

Example: university database

Physical schema:

- Relations stored as unordered files.
- Index on first column of students

Conceptual schema:

- Student (sid: string, name: string, age: number, percent: real)
- Courses (cid: string, cname: string, credits: number)
- Enrolled (sid: string, cid: string, grade: string)

External schema:

- Course_info(cid: string, enrollment: integer)

DATA INDEPENDENCE

DATA INDEPENDENCE

The changes can be made in one level without affecting the other levels that is called data independence.

Data independence is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

Types of data independence

Physical data independence is the capacity to change the internal schema without having to change the conceptual (or external) schemas.

Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs.

Entity- Relationship Model

Entity- Relationship Model

The E-R model is the most commonly used conceptual model.

In this model, the real world consists of a collection of basic objects called entities and the relationships among these objects.

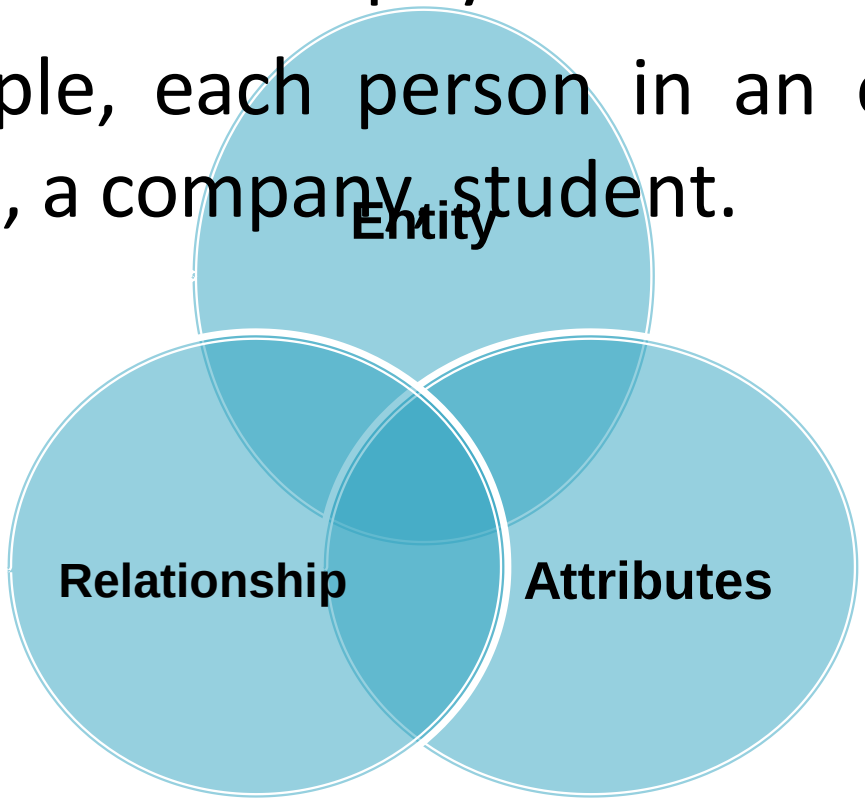
The end product of the modeling process is an entity-relationship diagram (ERD) or ER diagram.

But it is not implemented but design for creating the database.

The E-R data model employ three basic notions

Entity

- It is an object with a physical existence.
- For example, each person in an enterprise , car, house, a company, student.



Entity

It is an object with a physical existence.

For example, each person in an enterprise, car, house, a company, student.

Entity Type & Entity Sets

Entity Type – collection of entities that have the same attributes.

- Ex: STUDENT

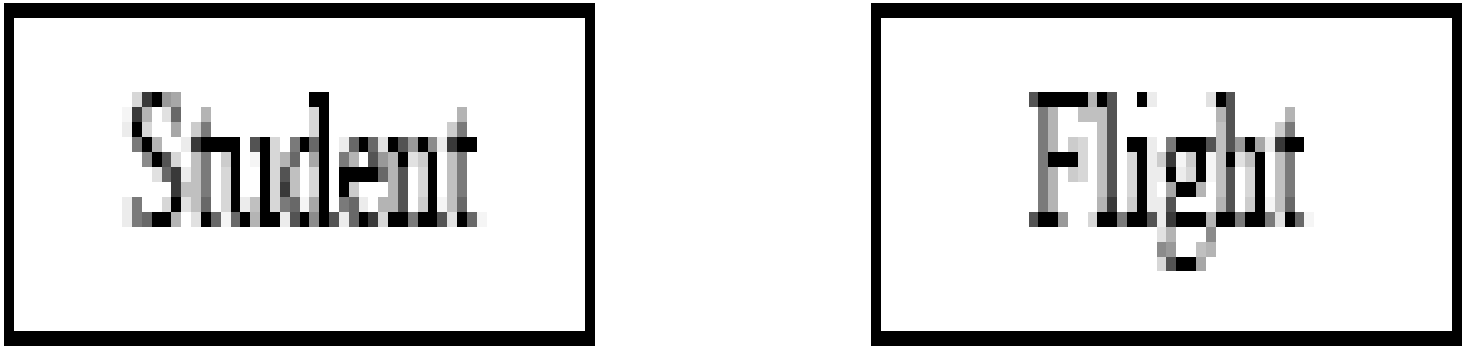
Entity Set – The collection of all entities of a particular entity type.

Ex: Set of all rows 10 rows of STUDENT

STUDENT

Name	Age	Rollno

Graphical representation of entity sets



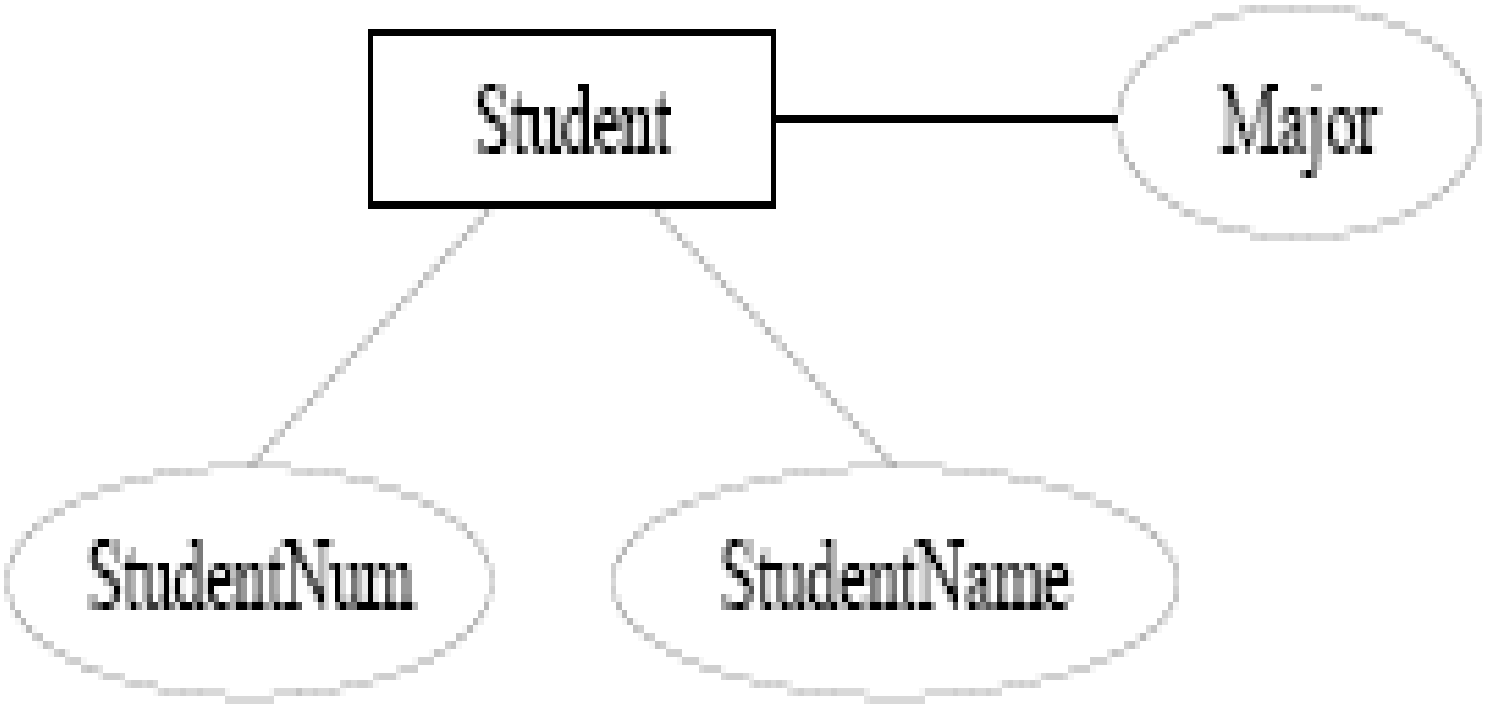
Attributes Closure

Attributes

Attributes are the particular properties that describe an entity.

Ex: A STUDENT entity may be described by student's name, student's roll_number.

Graphical representation of attributes



Types of Attributes

- Simple (Atomic) and Composite Attributes
- Single Valued & Multi-valued Attributes
- Stored and Derived Attributes
- Null Valued Attributes
- Complex Attributes

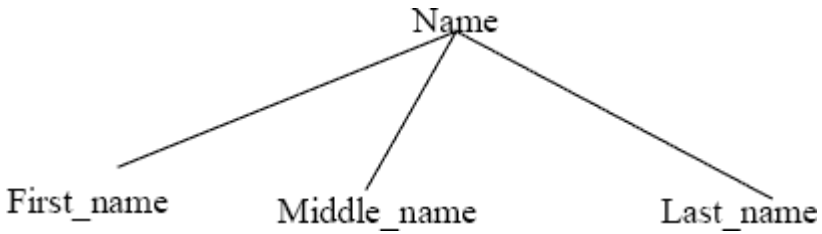
Simple (Atomic) and Composite Attributes

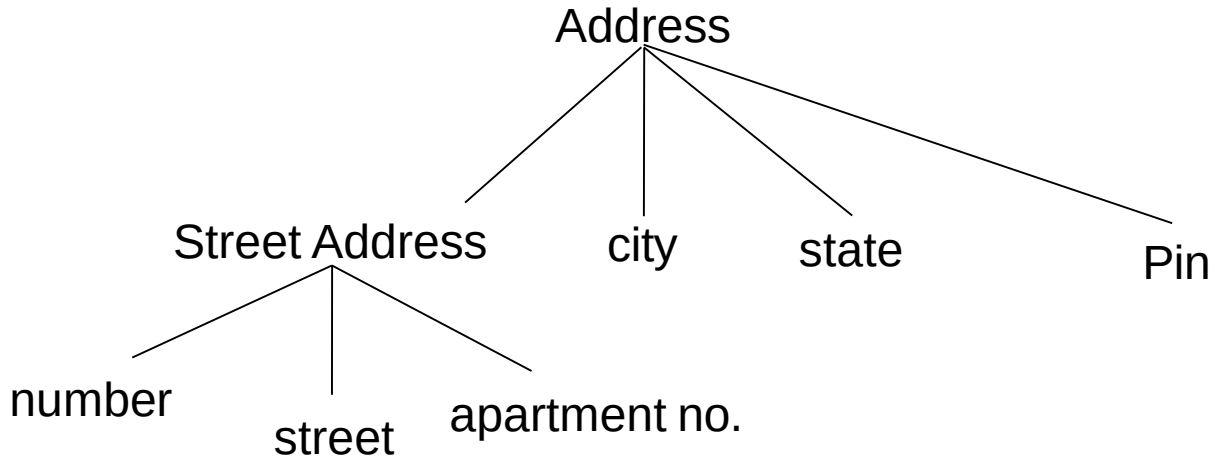
Simple attributes

- These are not divisible into parts.
- For example:
- Employee_Number and Age.

Composite attributes

- These can be divided into smaller subparts. These subparts represent basic attributes with independent meanings of their own.
- For example, take Name and address attributes.


Name
First_name
Middle_name
Last_name


Address
Street Address
city
state
Pin
number
street
apartment no.

Single Valued & Multi-valued Attributes

Single-valued attributes have a single value for particular entity.

Example: Roll_no, Age.

Multi-valued attributes may have more than one value for a single entity.

Example: Phone_no

Stored and Derived Attributes

Derived attribute is not stored in the database but it is derived from some attributes.

- Example: If DOB is stored in the database then we can calculate age of a student by subtracting DOB from current date.

Hence, in this case DOB is the stored attribute and age is considered as derived.

Null Valued Attributes

Null value is a value which is not inserted but it does not hold zero value.

The attributes which can have a null value called null valued attributes.

Example: Mobile_no attributes of a person may not be having mobile phones.

Complex Attributes

Complex composite attribute is a combination of and multi-valued attributes.

Complex attributes are represented by { } and composite attributes are represented by ( ).

Example1: Address_phone attribute will hold both the address and phone_no of any person.

Example2: {(2-A, St-5, Sec-4, Bhilai), 2398124}

Key attribute in an entity type

Key attributes will be having a unique value for each entity of that attribute.

It identifies every entity in the entity set. Key attribute will never be a null valued attribute.

Any composite attribute can also be a key attribute.

Example: roll_no, enrollment _no

There could be more than one key attributes for an entity type.

Domain of value set of an attribute

Domain of an attribute is the allowed set of values of that attribute.

Example: if attribute is ‘grade’, then its allowed values are A,B,C,F.

Grade ={A, B,C,F}

TYPES OF ENTITY TYPES

Weak entity type – Entity type that does not have any key attribute.

An entity in a weak entity type is identified by a relationship with a strong entity type and that relationship is called **Identifying Relationship** and that strong entity type is called the owner of the weak entity type.

TYPES OF ENTITY TYPES

Student

Roll No.	Name	Age
1	Rakesh	20
2	Nikhil	21
3	Nikhil	21

Secured

Marks

Name	M1	M2	M3
Nikhil	50	45	40
Nikhil	80	75	82

Identifying Relationship

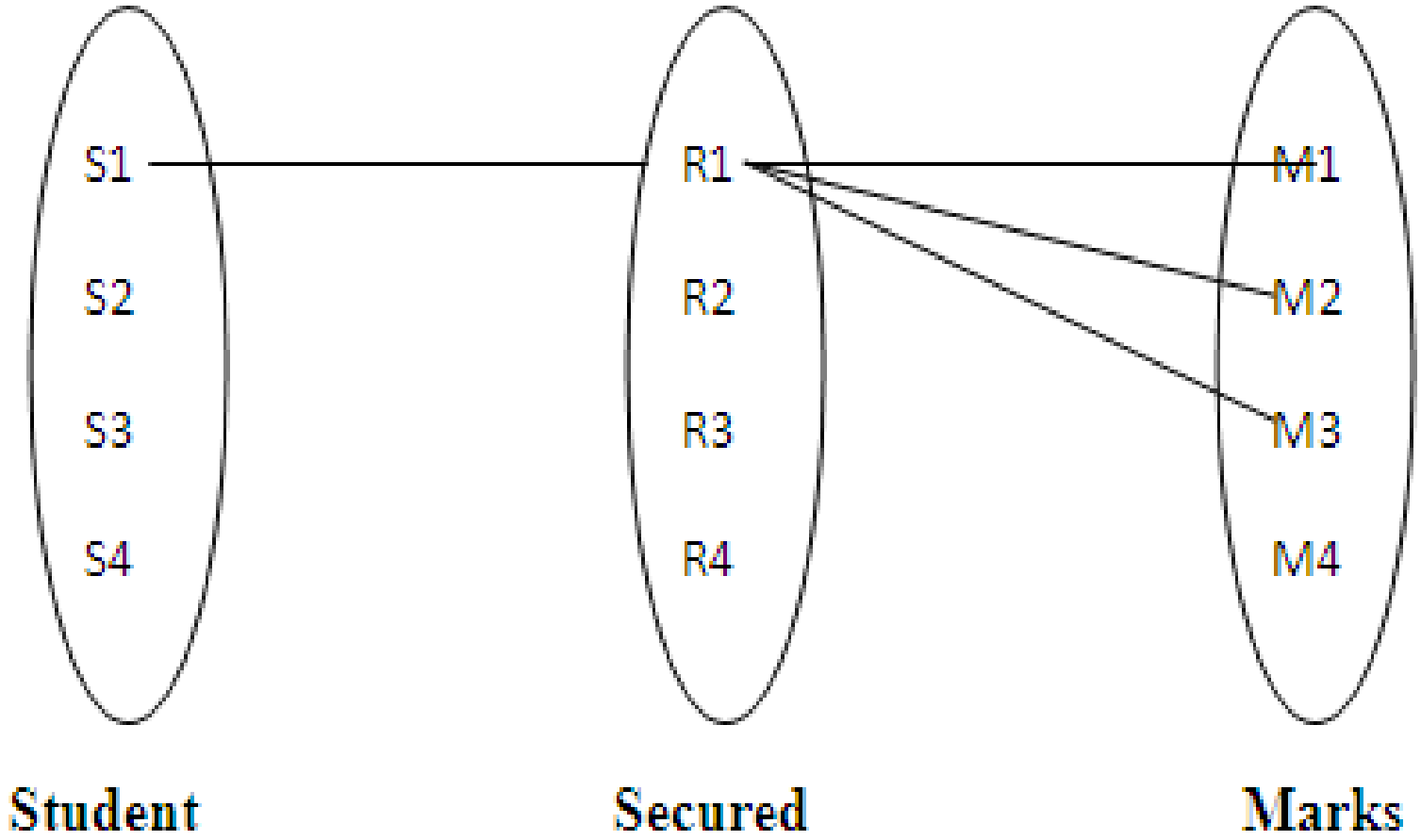
Relationship

Relationship

Relates two or more distinct entities with a specific meaning.

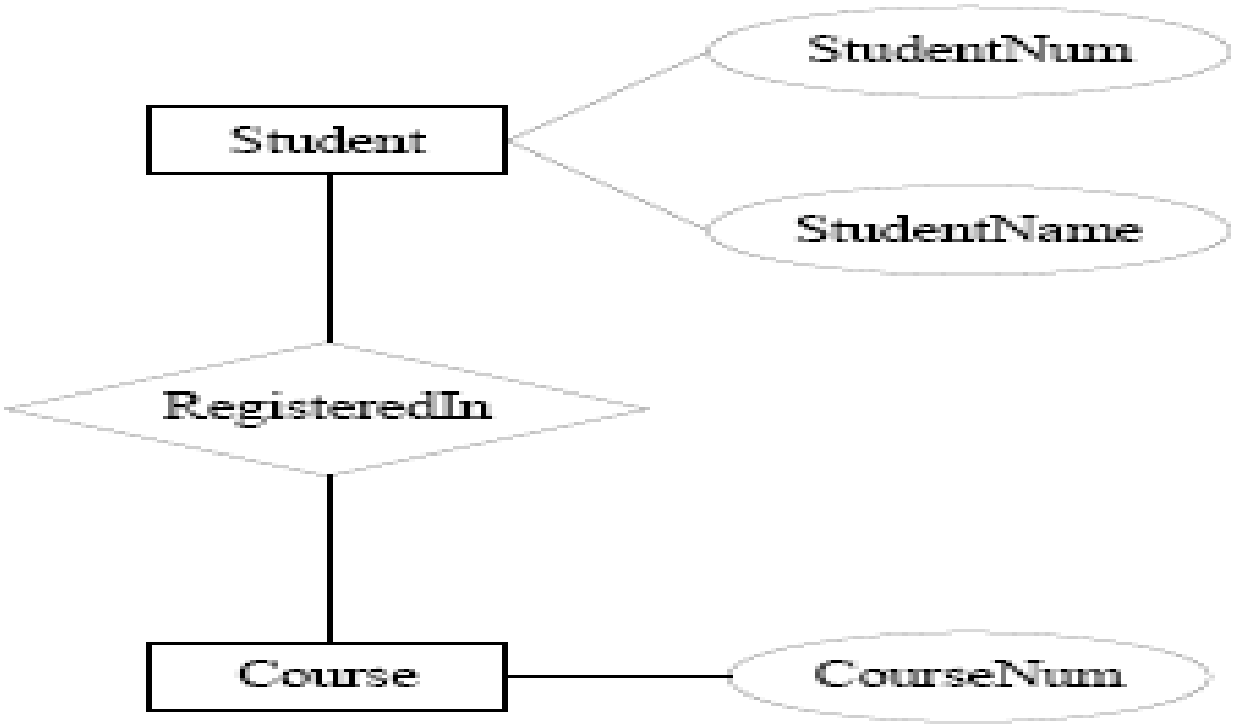
- For example, EMPLOYEE John works on the Product X PROJECT or EMPLOYEE Franklin manages the Research DEPARTMENT.

Terms used: Relationship type, Relationship set, Relationship instances.



Relationship type: secured
Relationship set: {R1, R2, R3, R4}
Relationship instances: R1

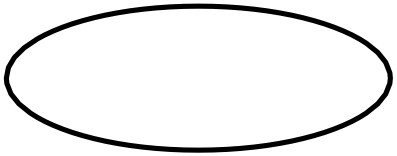
Graphical Representation of Relationship Sets



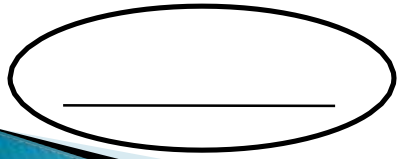
NOTATIONS USED IN E-R DIAGRAM

Entity Type

Weak Entity Type

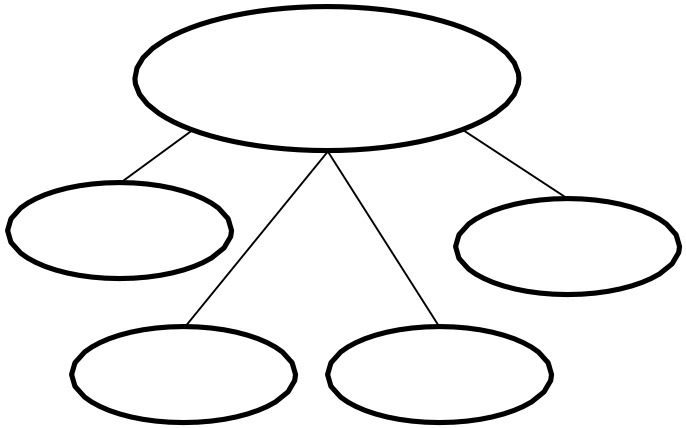


Attribute

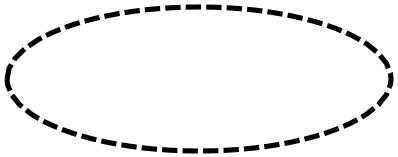


Key Attribute

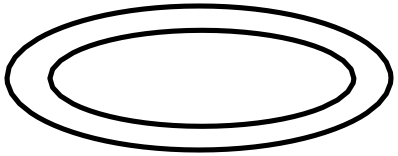
NOTATIONS USED IN E-R DIAGRAM



Composite Attribute

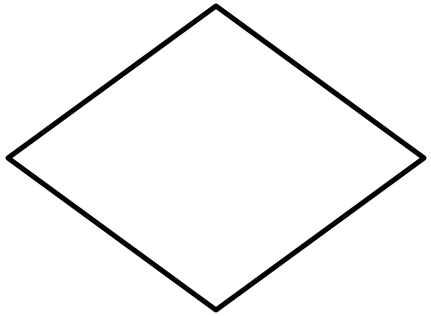


Derived Attribute

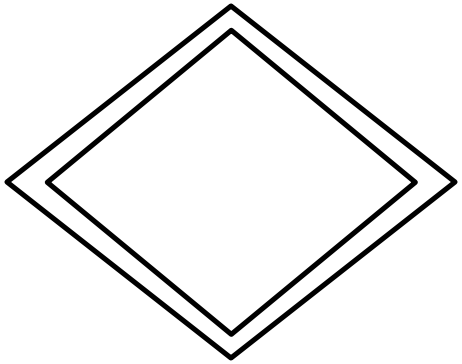


Multivalued Attribute

NOTATIONS USED IN E-R DIAGRAM



Relationship Type



Identifying Relationship

Constraints

Relationship types usually have certain Constraints.

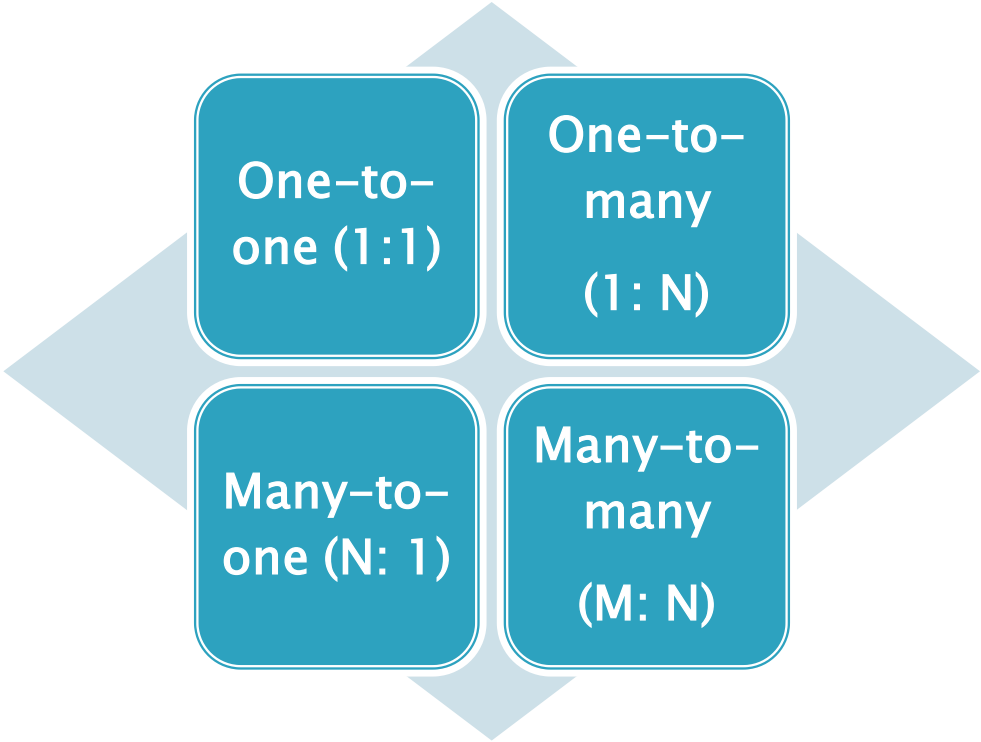
Two main types of relationship Constraints:

Mapping cardinalities
Participation constraints

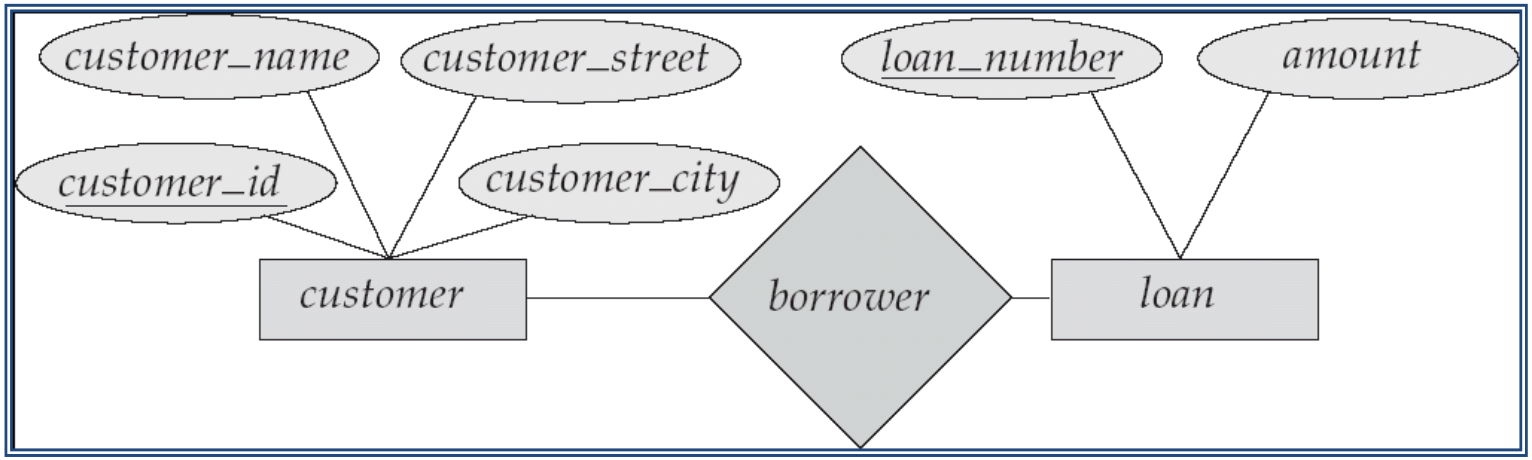
Mapping cardinalities, or cardinality ratios

Specifies the number of relationship instances that an entity can participate in.

Mapping Cardinalities

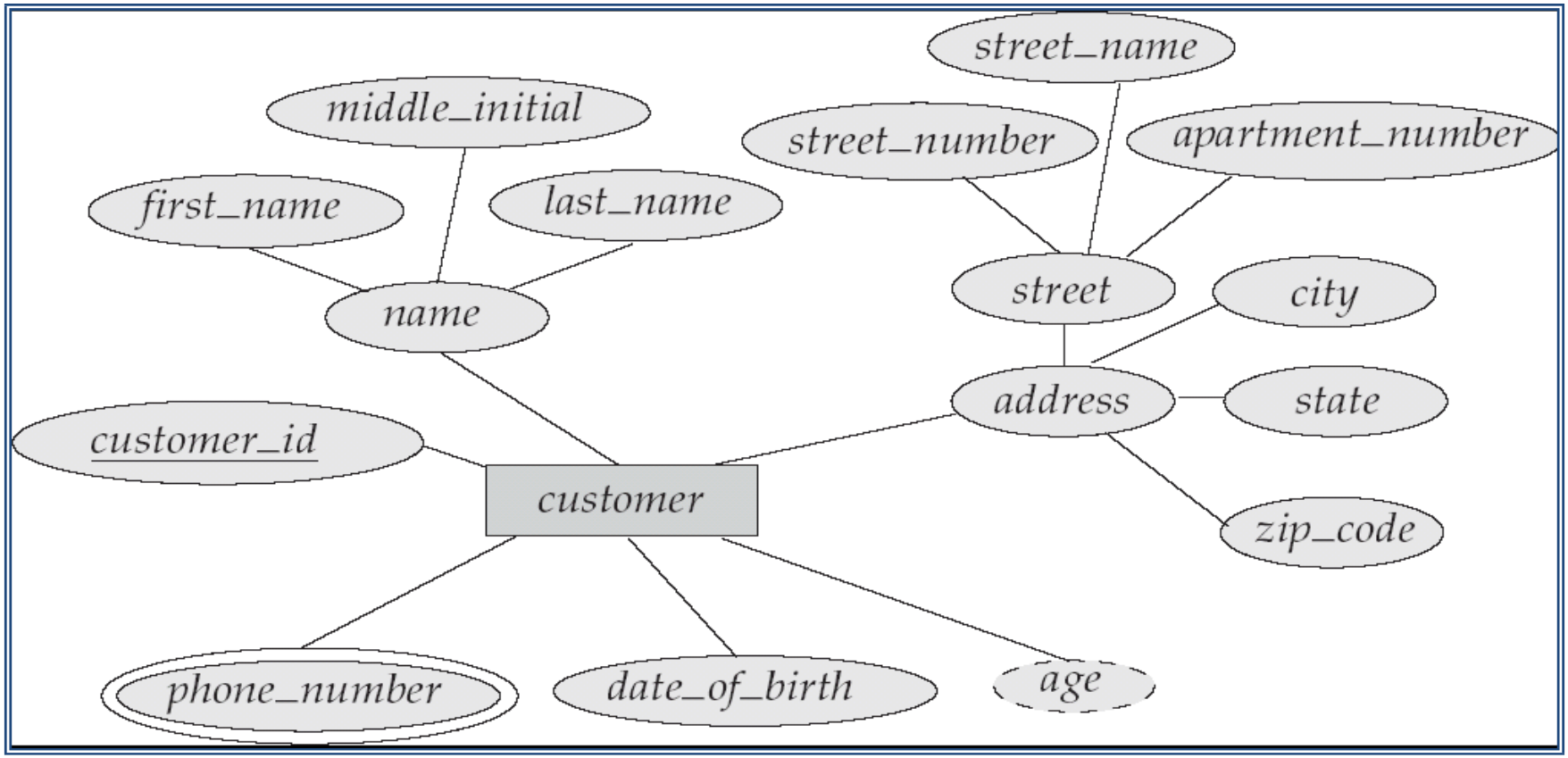


Example of E-R Diagrams

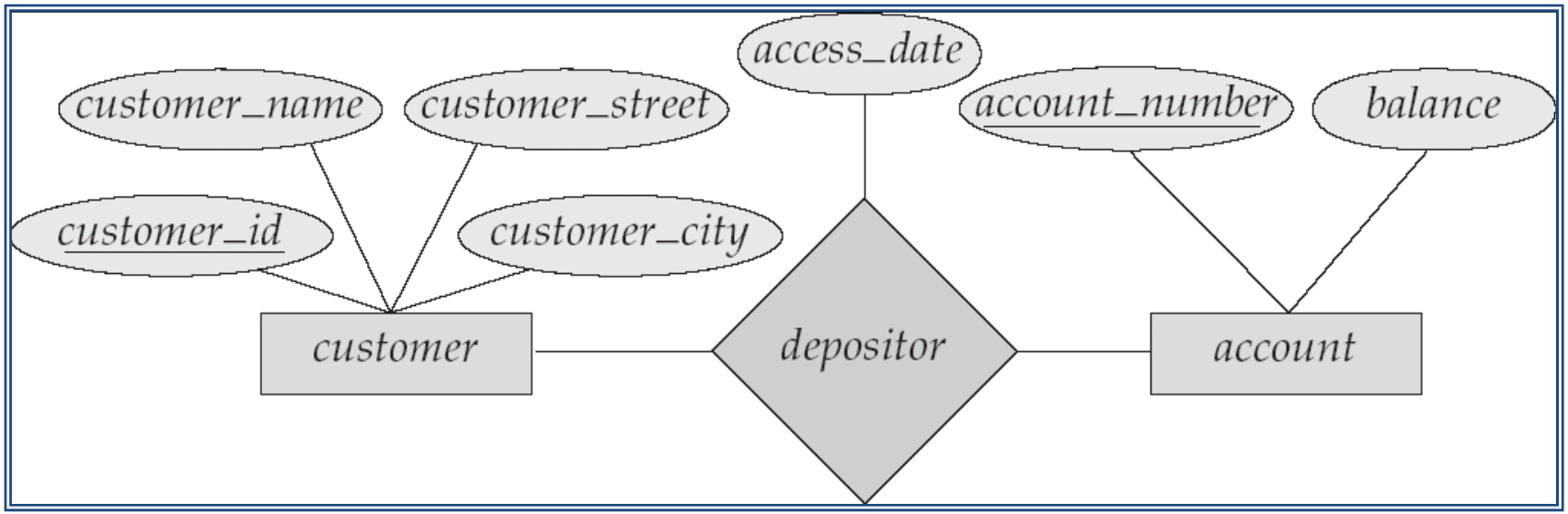


- Rectangles represent entity types.
- Diamonds represent relationship types.
- Lines link attributes to entity types and entity types to relationship types.
- Ellipses represent attributes
- Underline indicates primary key attributes (will study later)

E-R Diagram With Composite, Multivalued, and Derived Attributes



Relationship Types with Attributes



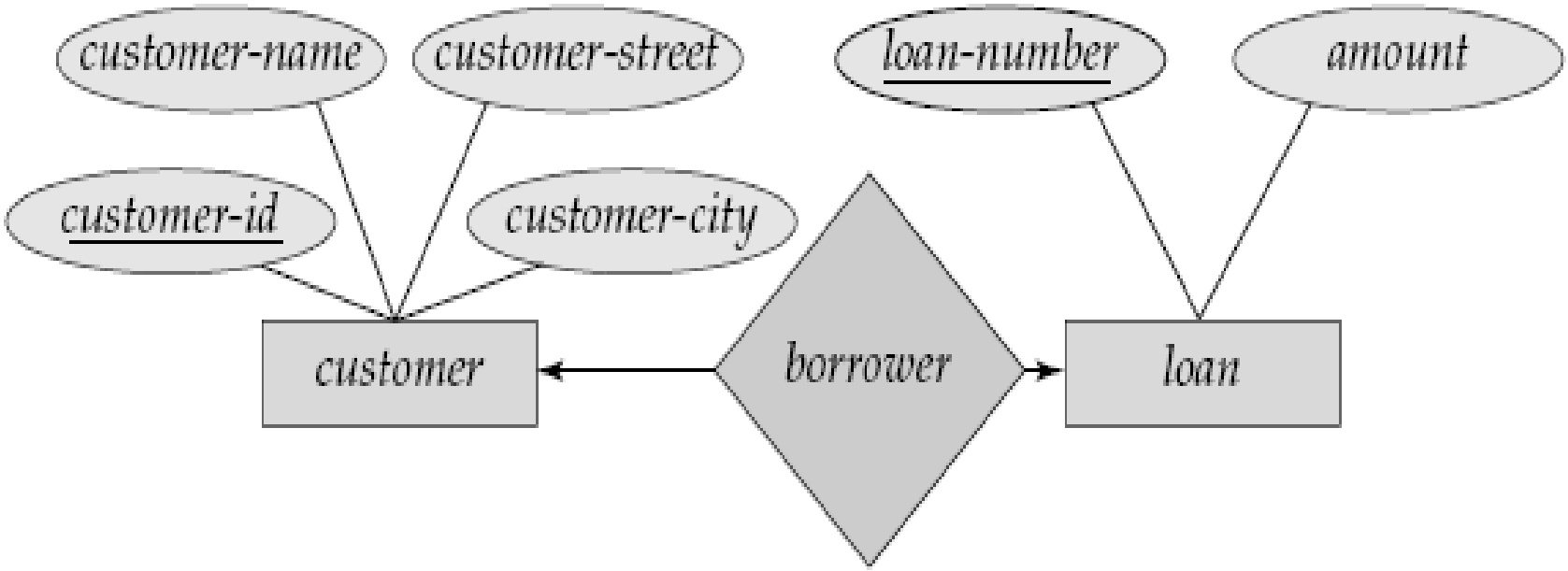
we have the access_date attribute attached to the relationship set depositor to specify the most recent date on which a customer accessed that account.

Cardinality ratio

We express cardinality ratio by drawing

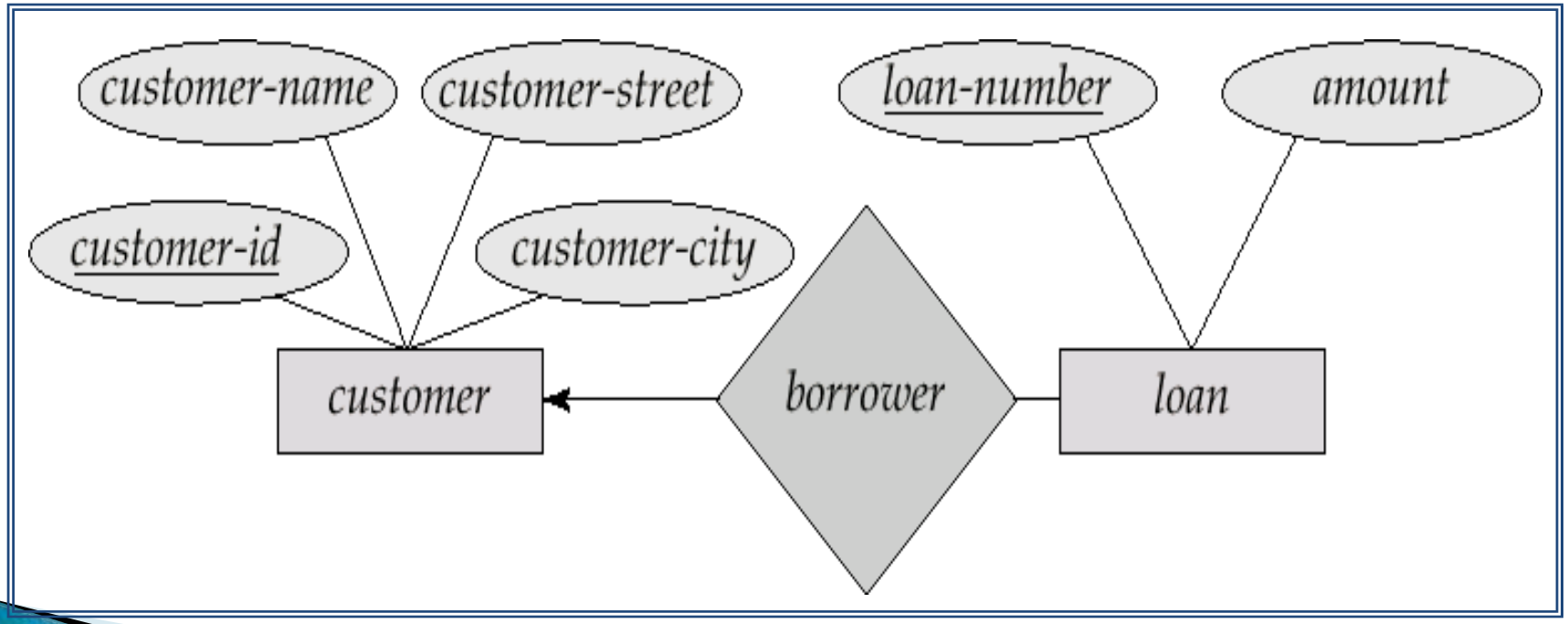
- directed line (→), signifying “one,” or an
- undirected line (—), signifying “many,”

One-To-One Relationship



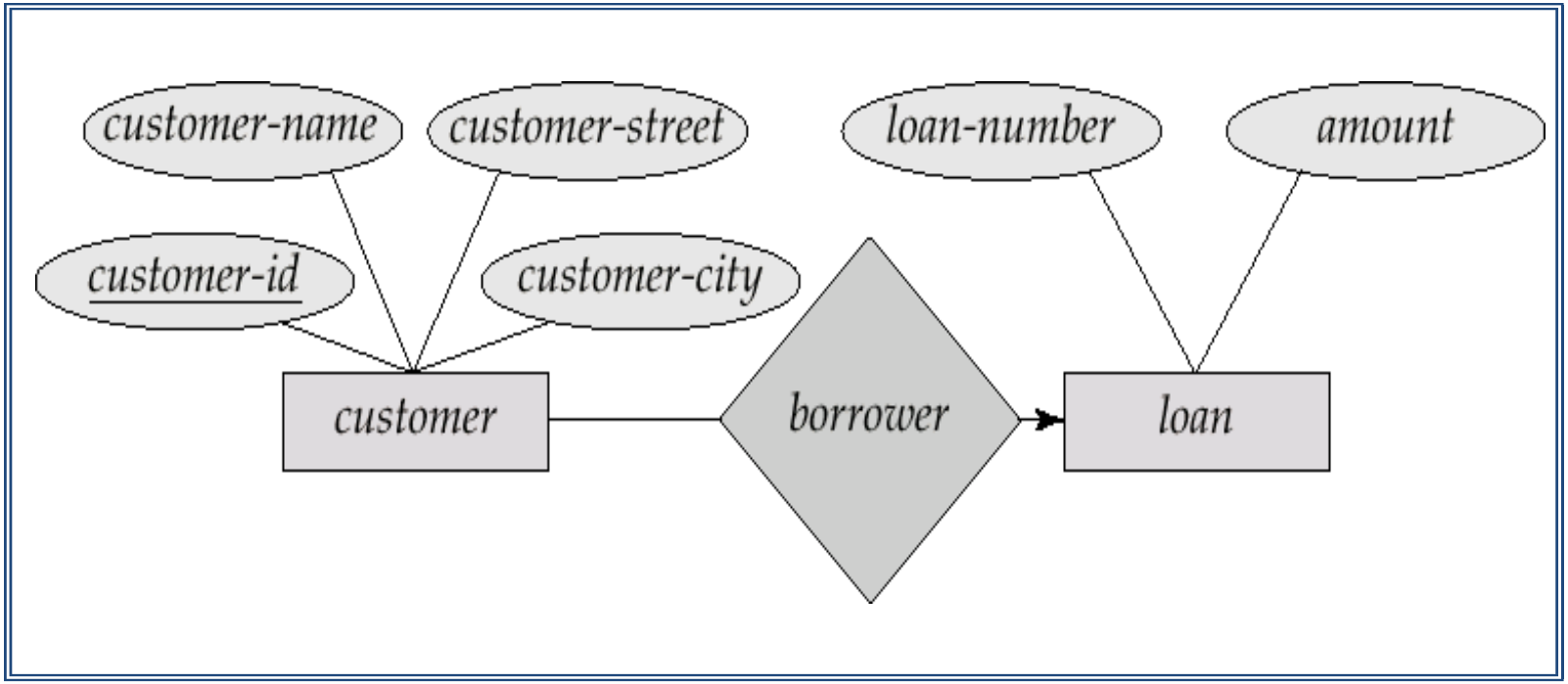
One-To-Many Relationship

In the one-to-many relationship a customer is associated with several loans via *borrower*



Many-To-One Relationships

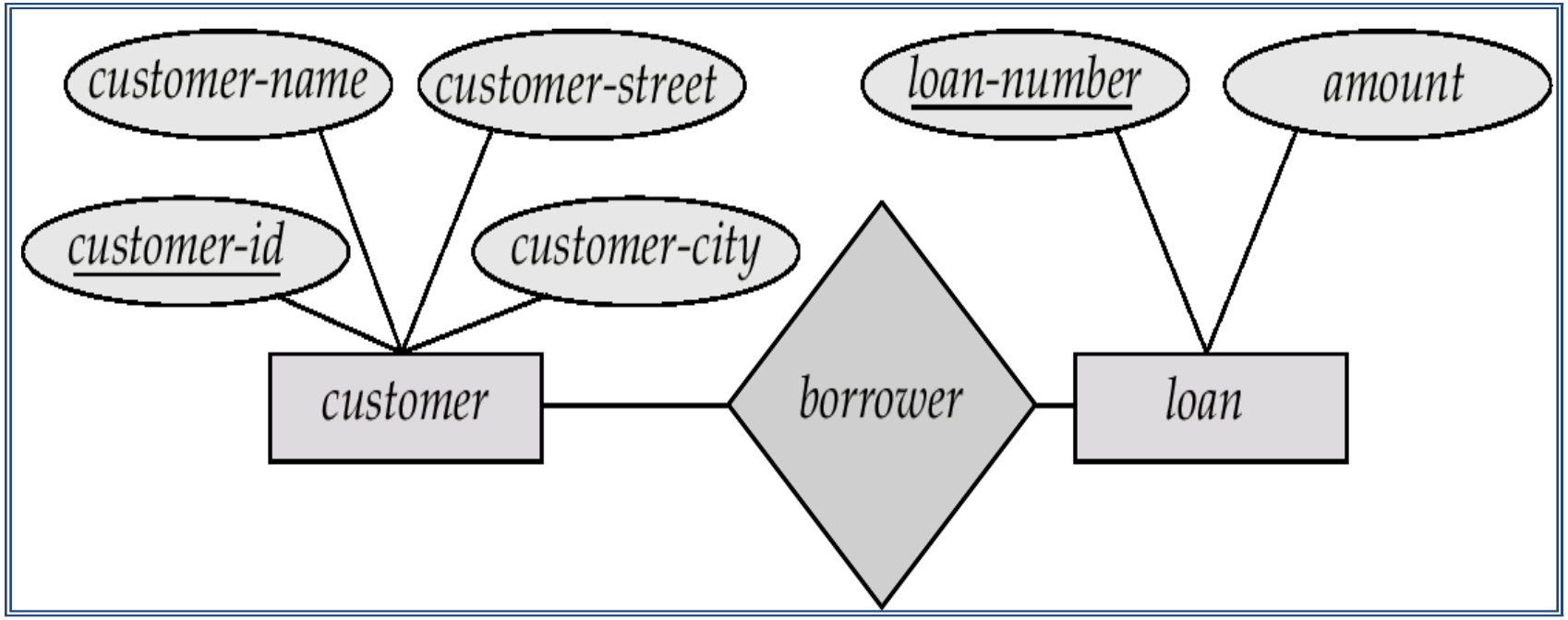
In a many-to-one relationship a loan is associated with several customers via *borrower.*



Many-To-Many Relationship

A customer is associated with several (possibly 0) loans via borrower

A loan is associated with several (possibly 0) customers via borrower

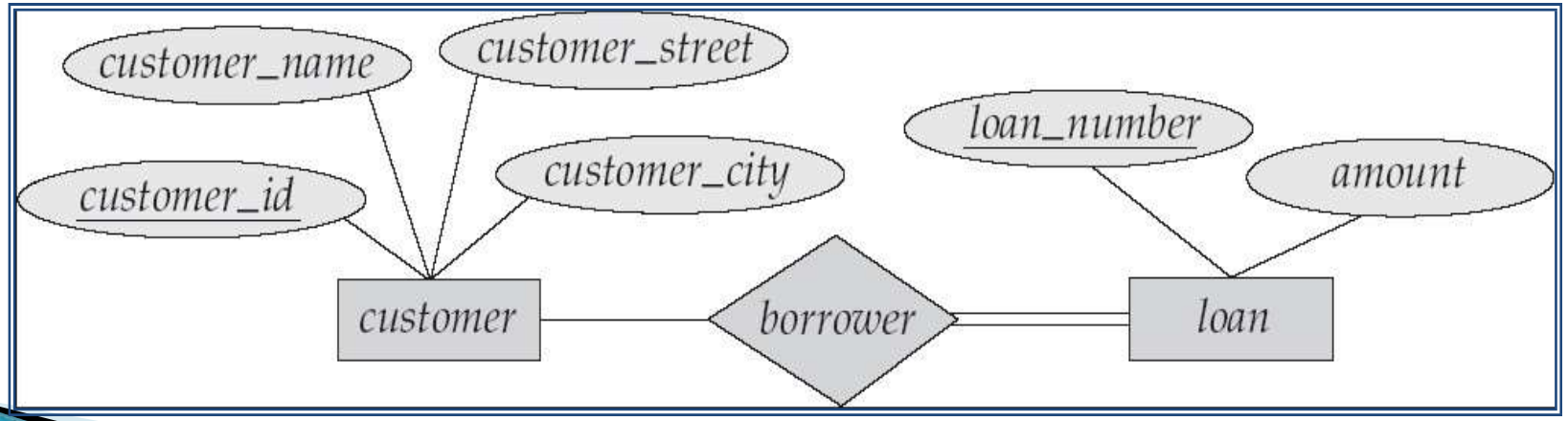


Find out the Cardinality ratio

- Prime minister-country
- classroom –students
- students –classroom
- customer -loan

Participation constraints

- **Total participation :** every entity in the entity type participates in at least one relationship in the relationship type
  - E.g. participation of loan in borrower is total
    - every loan must have a customer associated to it via borrower
- **Partial participation**: some entities may not participate in any relationship in the relationship type
  - Example: participation of customer in borrower is partial
    - some customers may not participate in any loan

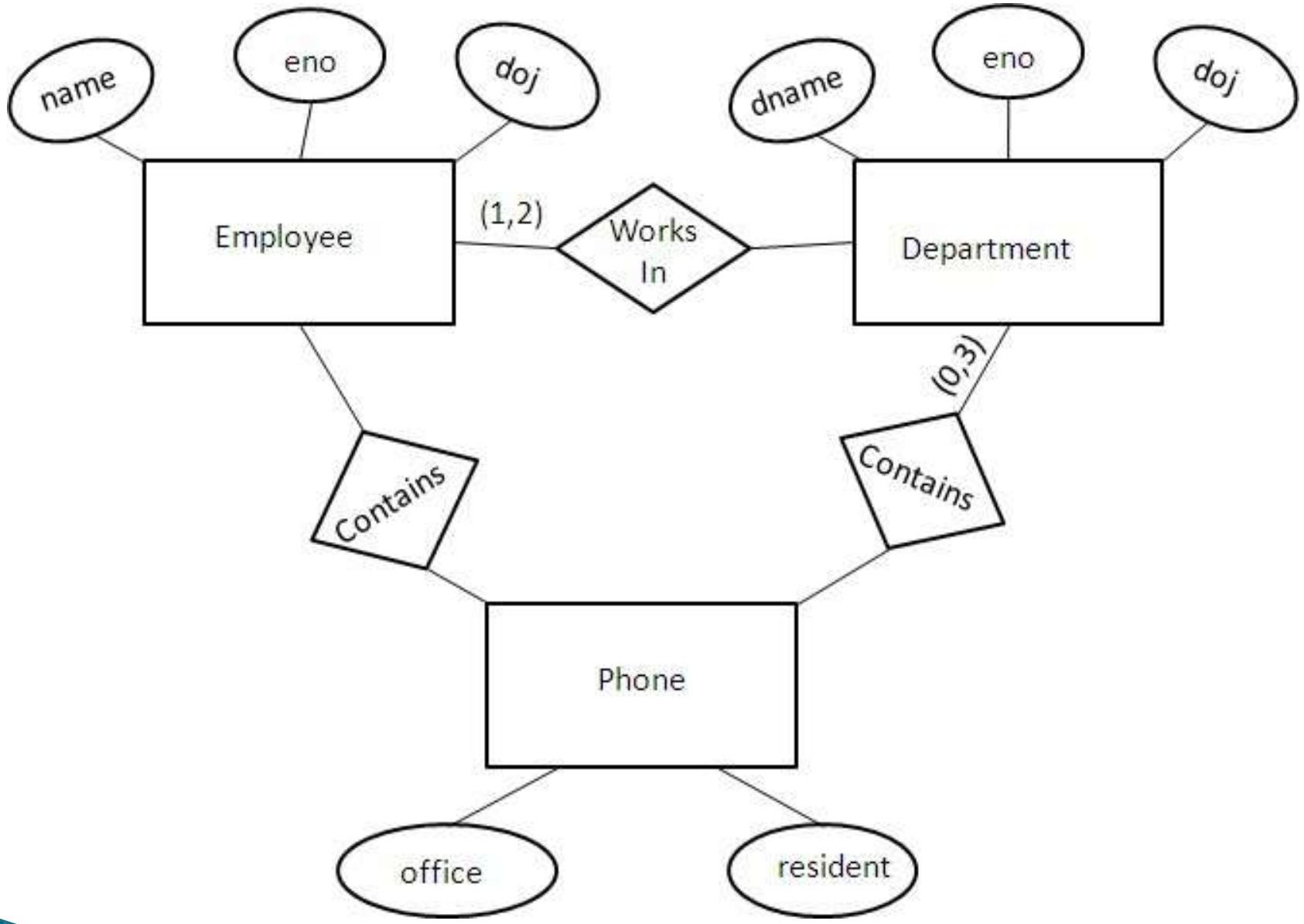


Steps in ER Modeling

- Identify the Entities
- Find relationships
- Identify the key attributes for every Entity
- Identify other relevant attributes
- Draw complete E-R diagram with all attributes including Primary Key

PROBLEMS ON E-R DIAGRAM

Question: An employee works in one department. The department contains phone, the employee also has phone. Assume that an employee works in maximum 2 departments or minimum one department. Each department must have maximum 3 phones or minimum zero phone. Design an E-R diagram for the above.


name
eno
doj
dname
eno
doj
Employee
(1,2)
Works In
Department
(0,3)
Contains
Contains
Phone
office
resident

EER (Enhanced Entity-Relationship)

- The EER model is a high-level or conceptual data model incorporating extensions to the original Entity- relationship (ER) model.
- EER includes all the concepts of ER model.
- EER=ER all the concepts + some extension
- Additionally it includes the concepts of
  - superclass and subclass
  - specialization and generalization.

Subclasses and Superclasses

An entity type may have additional meaningful subgroupings.

Superclass

SECRETARY
ENGINEER
MANAGER
TECHNICIAN

SALARIED_EMPLOYEE
HOURLY_EMPLOYEE

Subclasses

Subclasses and Inheritance

- An entity class B is said to be a subclass of another entity class A. if it shares an "is-a" relationship with A.

- Example 1: Car "is-a" Vehicle
- Example 2: Manager "is-a" Employee

- An entity of class B is said to be a **specialization** of entities of class A.
- Conversely, entities of class A are **generalizations** of class B entities.

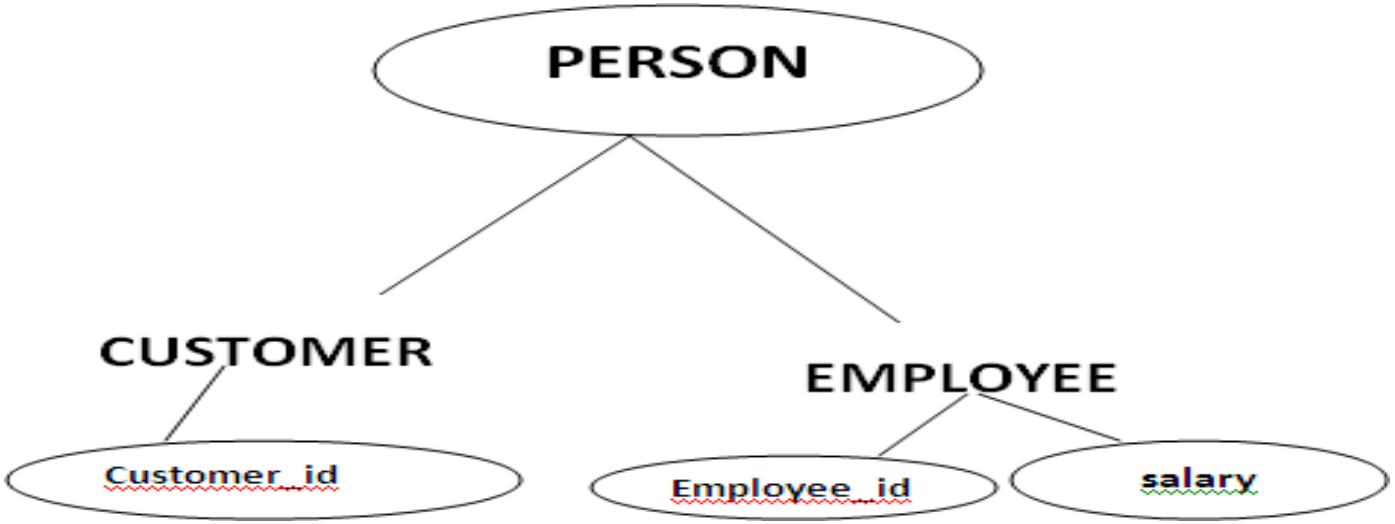
Specialization

Specialization is the process of defining a set of subclasses of a superclass. The set of subclasses is based upon some characteristics of the entities in the superclass.

It follows **top-down design** process.

Represented by a triangle component labeled ISA (E.g. customer "is a" person).

Example of Specialization



The specialization of person allows us to distinguish among persons according to whether they are employees or customers.

Generalization

Generalization is a simple inversion of specialization.

It is a **bottom-up** design process.

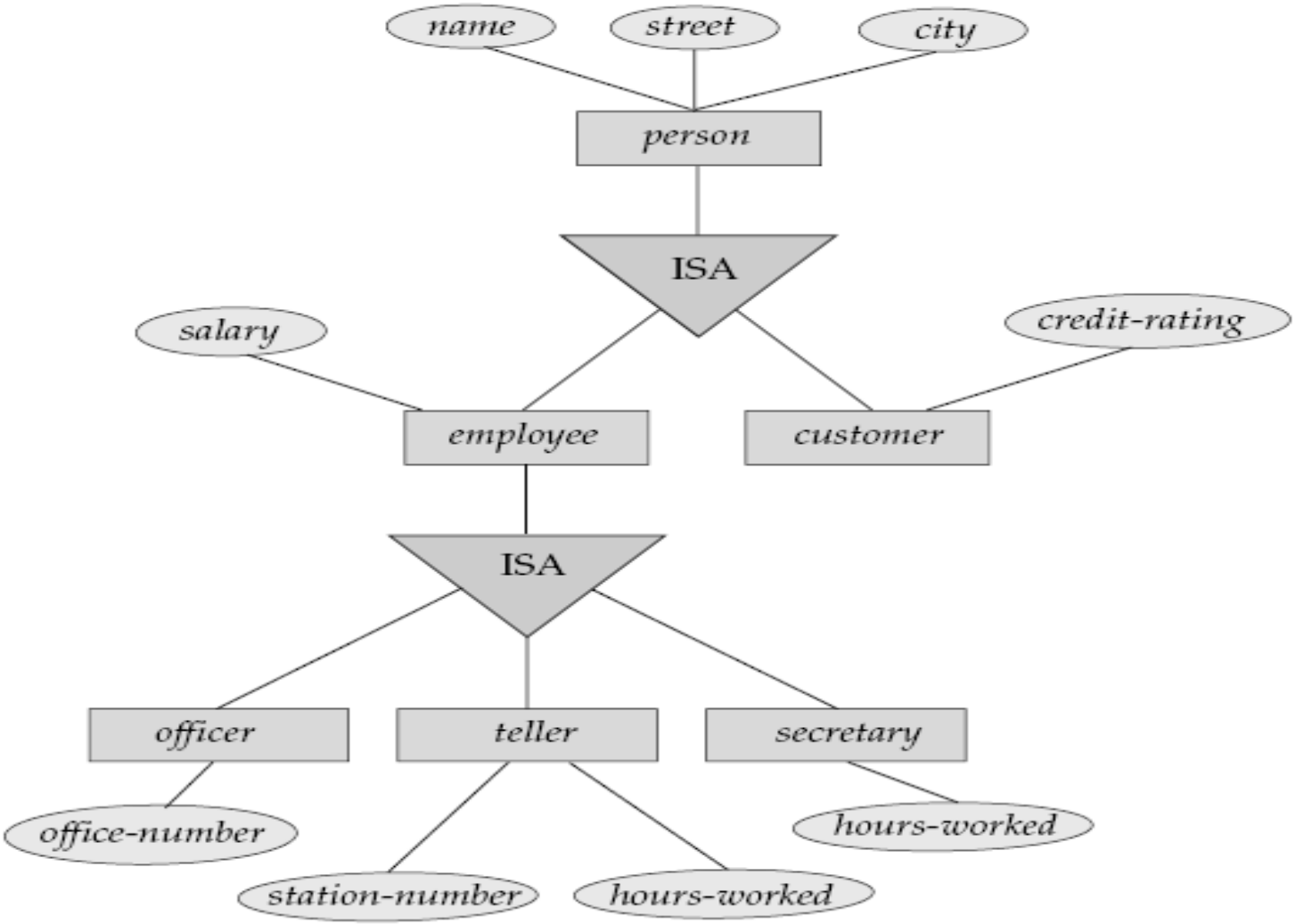
In this process multiple entity sets are synthesized into a higher-level entity set on the basis of common features.

For example, customer entity set with the attributes name, street, city, and customer-id, and an employee entity set with the attributes name, street, city, employee-id, and salary.

There are similarities between the customer entity set and the employee entity set in the sense that they have several attributes in common.

Differences in the two approaches may be characterized by their starting point and overall goal.

Specialization and generalization



Design Constraints on a Specialization/Generalization

Constraint on which entities can be members of a given lower-level entity set.

- **Condition-defined**
  - **User-defined**

Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.

- **Disjoint**
- **Overlapping**

Design Constraints on a Specialization/Generalization (Cont.)

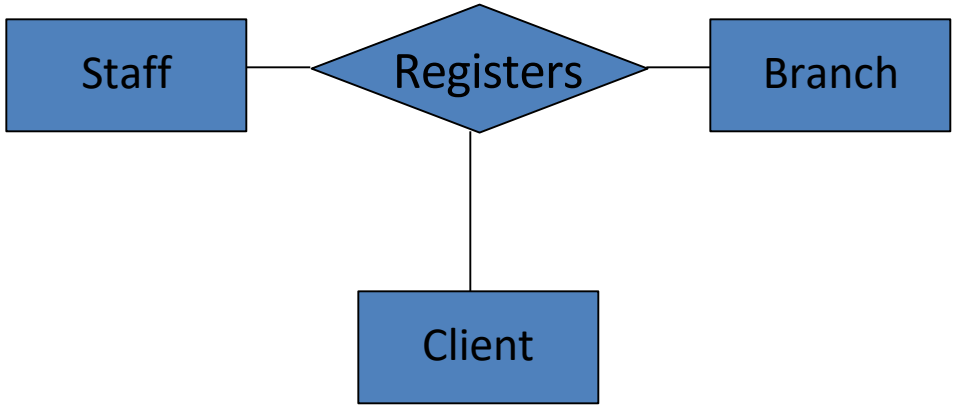
Completeness constraint -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.

- **total** : an entity must belong to one of the lower-level entity sets. The account generalization is total: All account entities must be either a savings account or a checking account.

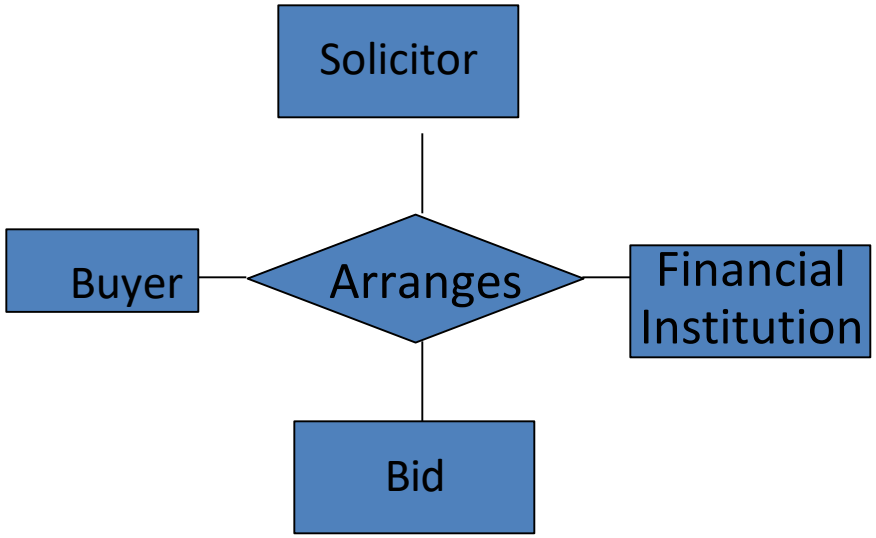
- **partial**: an entity need not belong to one of the lower- level entity sets. Since employees are assigned to a team only after 3 months on the job, some employee entities may not be members of any of the lower-level team entity sets.

Degree of Relationship Type

- It is the number of participating entity types in a relationship.
- A relationship of degree two is called **binary**, a relationship of degree three is called **ternary**...

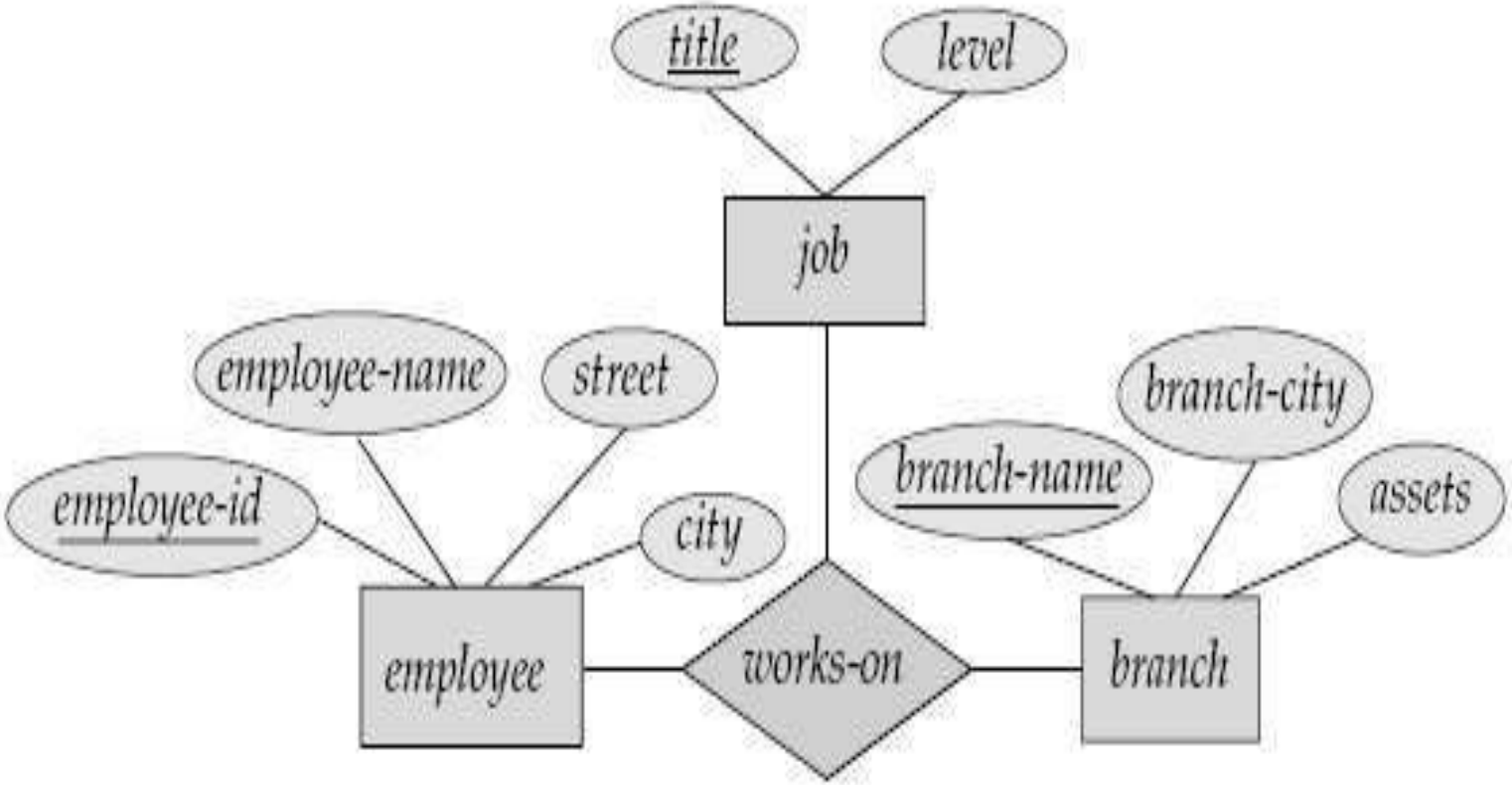


"Staff registers a client at a branch"

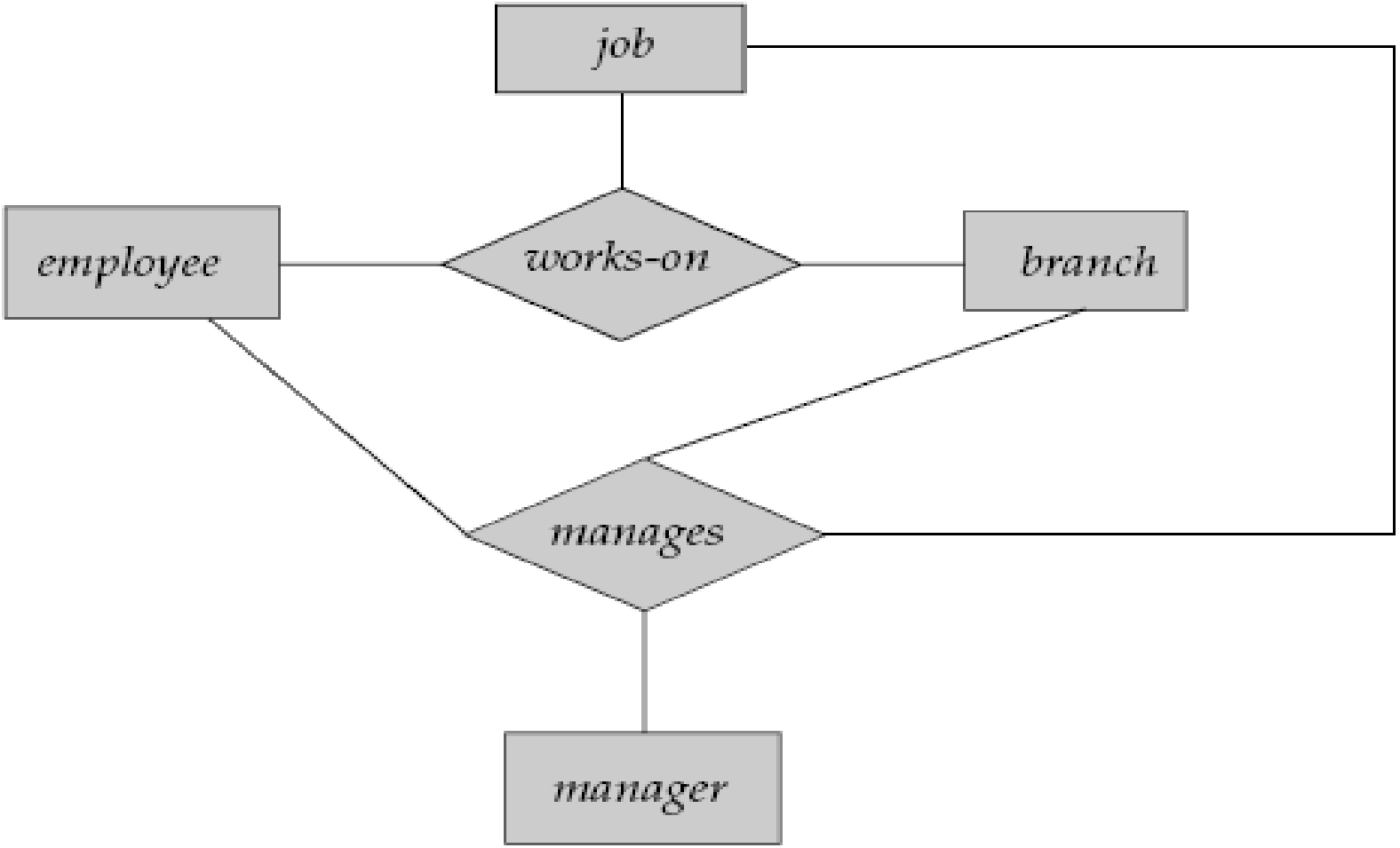


"A solicitor arranges a bid on behalf of a buyer supported by a financial institution"

A ternary relationship



E-R diagram with redundant relationships



Aggregation

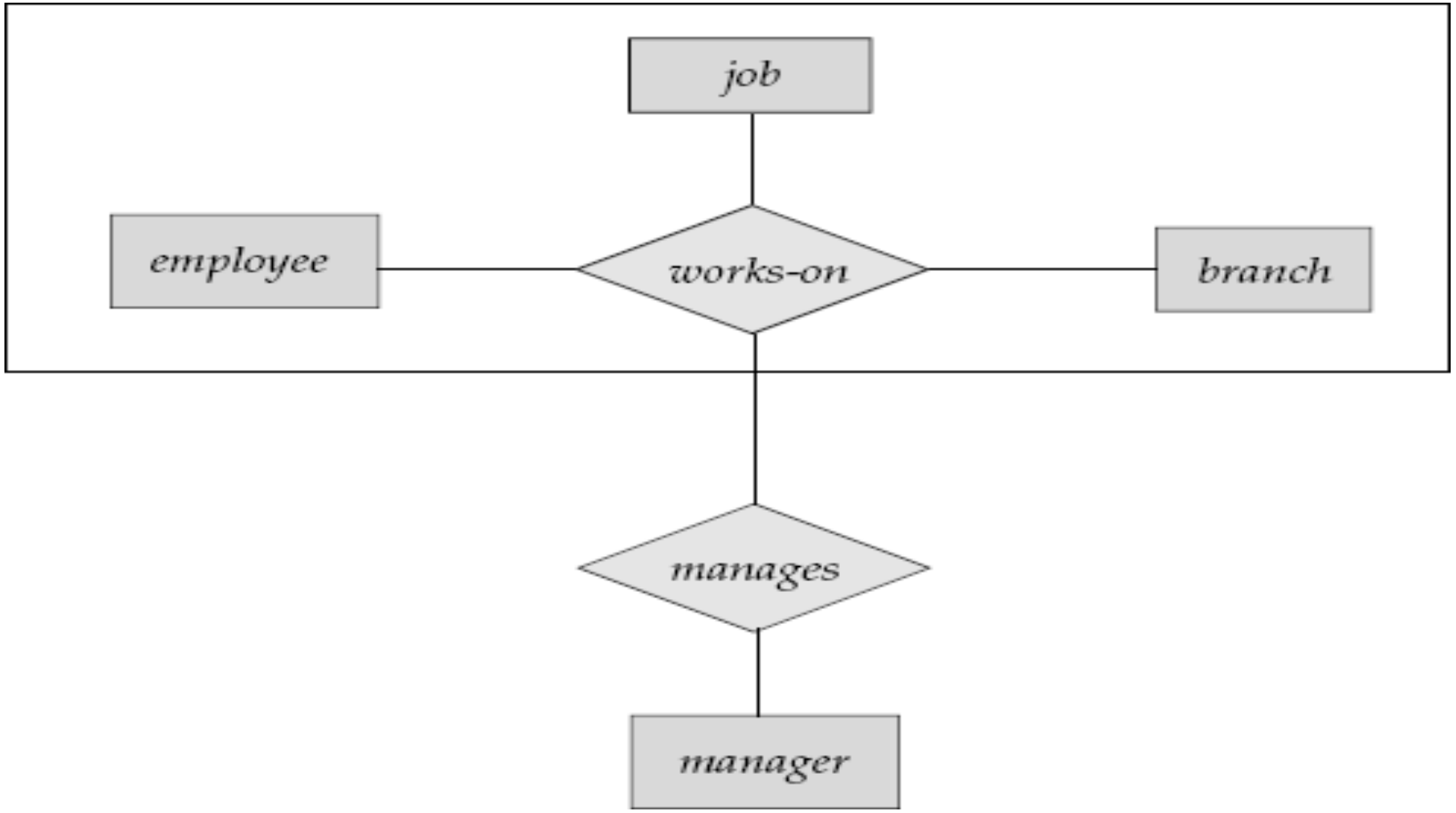
Aggregation is an abstraction through which relationships are treated as higher level entities.

- Thus, for our example, we regard the relationship set works-on (relating the entity sets employee, branch, and job) as a higher-level entity set called works-on.

Such an entity set is treated in the same manner as is any other entity set.

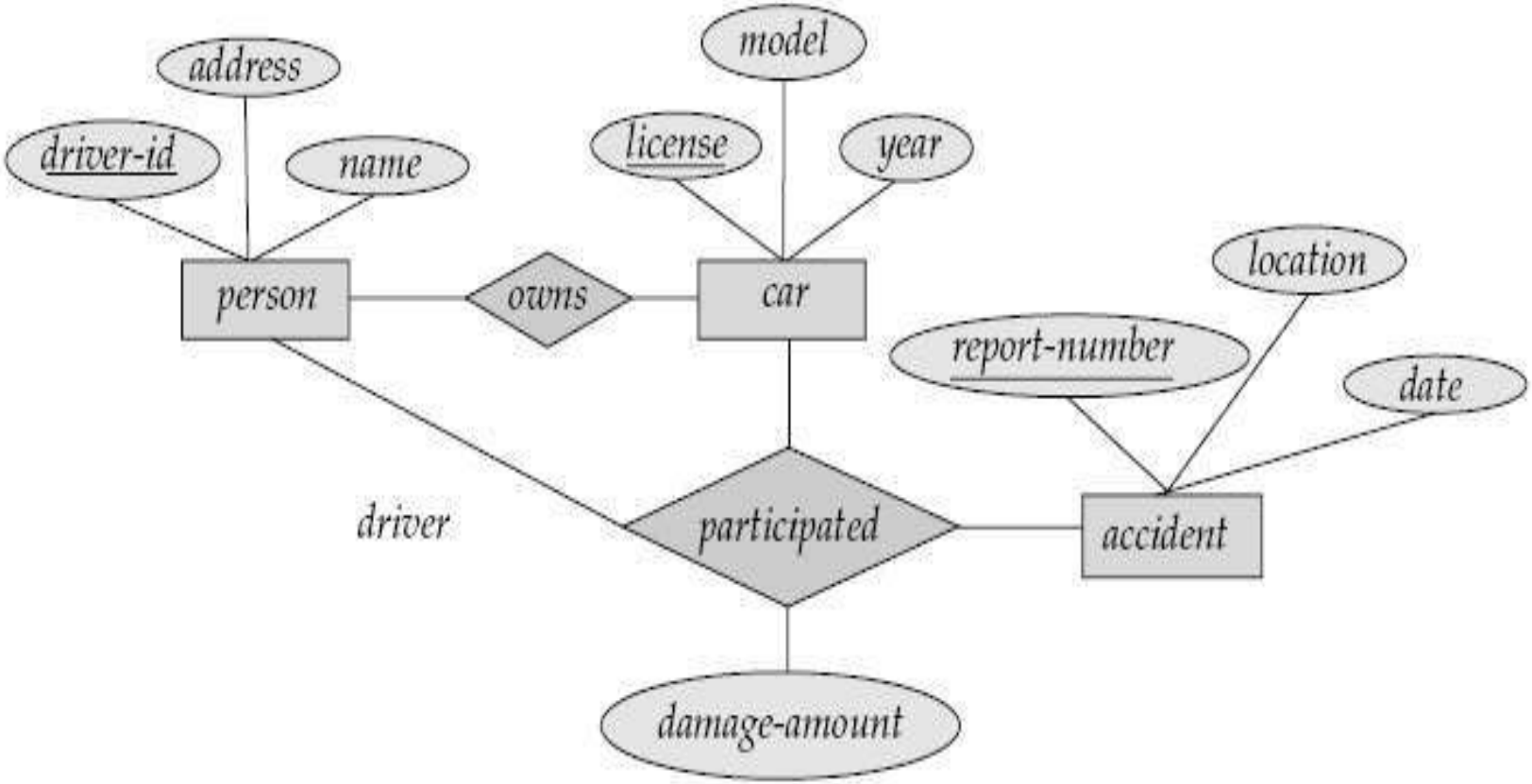
We can then create a binary relationship manages between works-on and manager to represent who manages what tasks.

E-R Diagram With Aggregation



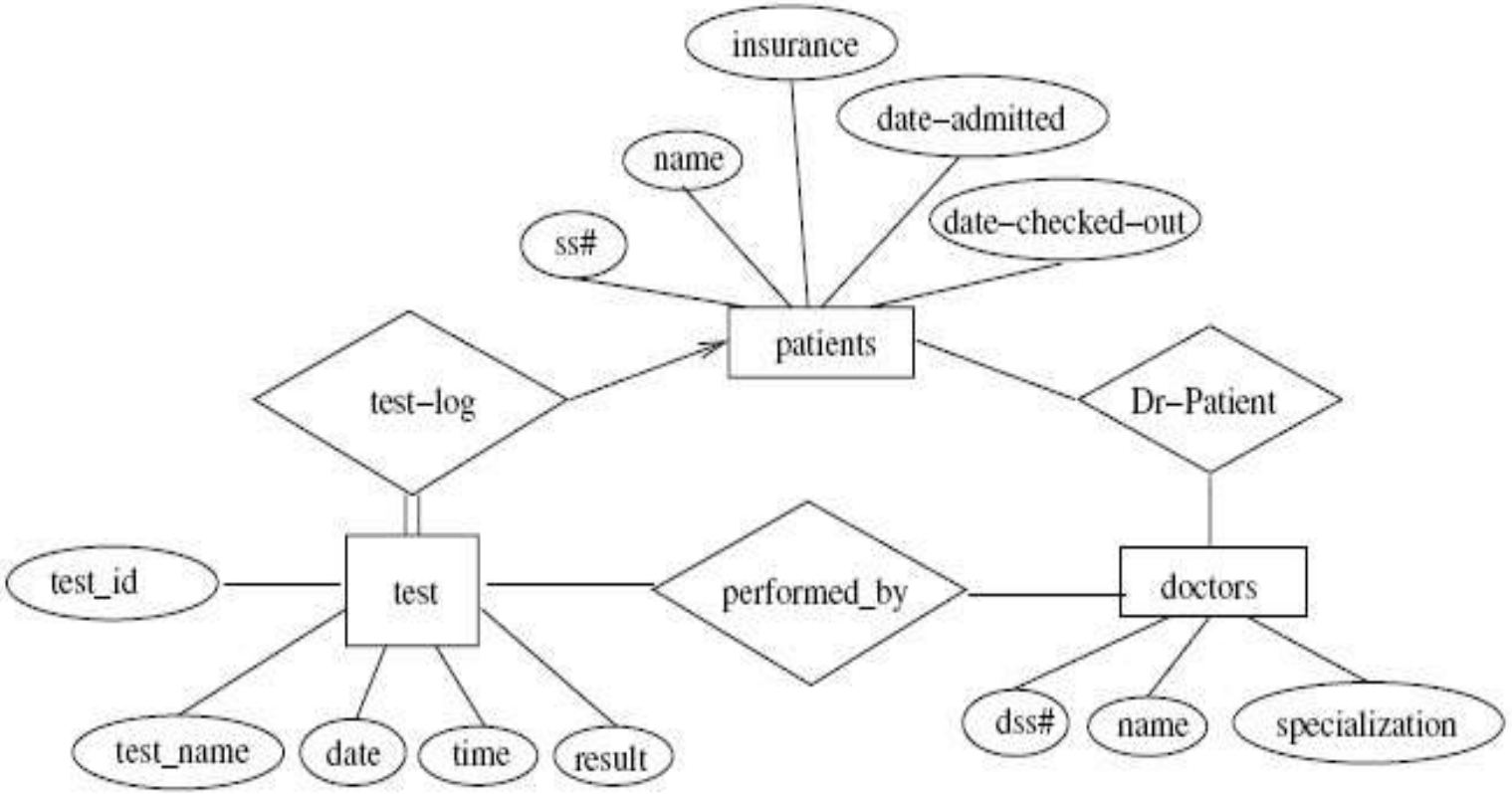
Question 1

Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.



Question 2

Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.



ER model question 1

Data in the database of a department store is as follows:

- Each employee is represented. The data about an employee are his employee number, name, address and the department he works for.
- Each department is represented. The data about departments are its name, employees, manager and items sold.
- Each item sold is represented. The data about items are its name, manufacturer, price, model number (assigned by the manufacturer) and an internal item number (assigned by the store).
- Each manufacturer is represented. The data about a manufacturer are its name, address, items supplied to the store and their prices.
- Give an ER diagram for this database. Indicate a key for each entity set and indicate which reips e-one which are many-one.

ER model question 2

- In university database, an entity type section which describes the section offering course.
- The attribute of e section are section number, semester, year, course number, instructor, room no, weekday(domain is the possible combinations of weekday in which the sections are offered {9-9.50 AM, 10-10.50AM,……3.30-4.50, 5.30-6.20PM etc}.
- assume that section number is unique for each course within a particular semester/year combination. There are several composite keys for section and some attributes are components of more than one key.
- Identify three composite keys, and show how they can be represented in an ER diagram schema.

ER model question 3

Create an ER diagram for the following:

- Each company operates four departments, and each department belongs to one company.
- Each department employs one or more employees and each employee work for one department.
- Each of the employee may or may not have one or more departments and each departments belong to one employee.
- Each employee may or may not have an employment history.
- Make assumptions if necessary and state them. Identify primary, candidate and foreign key.