

Databases 2022

Darko Bozhinoski,
Ph.D. in Computer Science

Agenda

- Recap
- ER Model (continue)
- ER advanced concepts
- ER vs. UML

Conceptual Data Models for Database Design

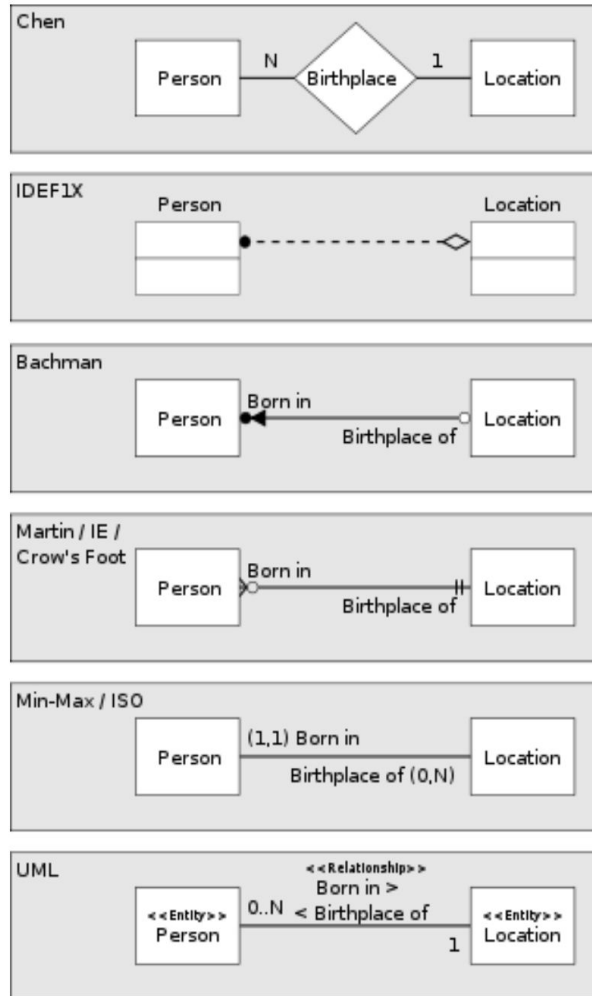
Entity - Relationship (ER) model:

- Popular high-level conceptual data model

ER diagrams:

- Diagrammatic notation associated with the ER model
- There are also slightly different notations

ER Notations



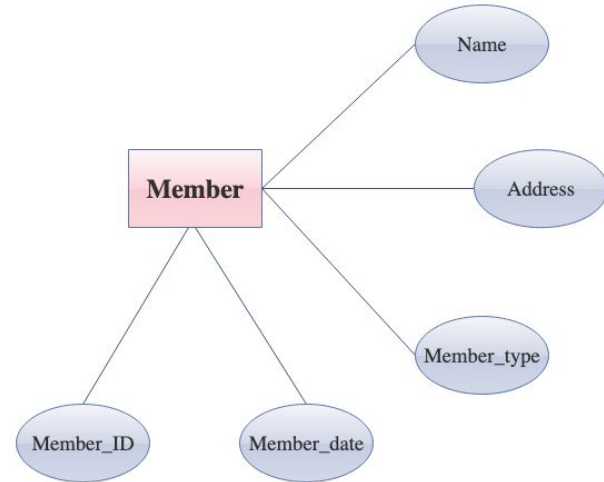
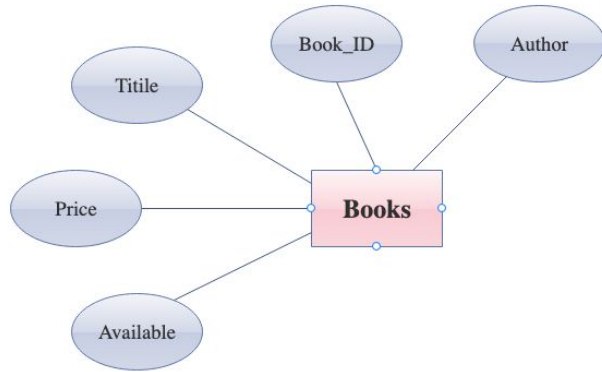
Entity-Relationship Model

Basic concepts

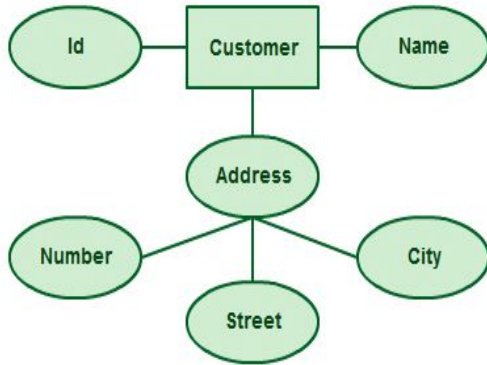
- **Entity:** corresponds to a thing or object in the real world
- **Relationship:** corresponds to a link between entities
- **Attribute:** a property that describes an entity

ER diagrams

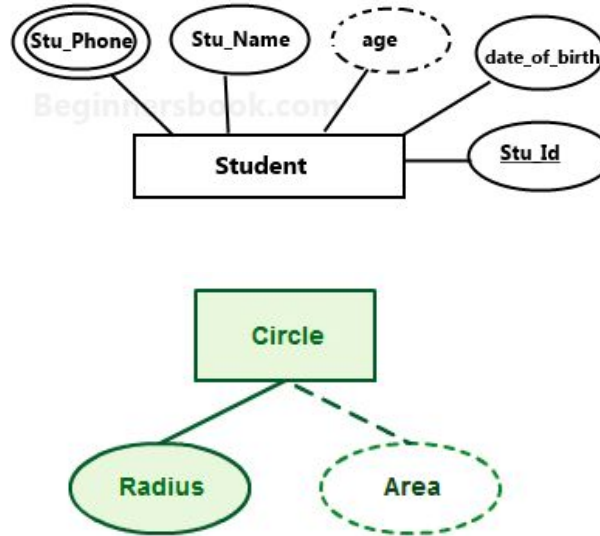
Entities have set a set of attributes.



ER diagrams: Attribute Types



Composite vs. atomic attributes



Stored vs. Derived attributes

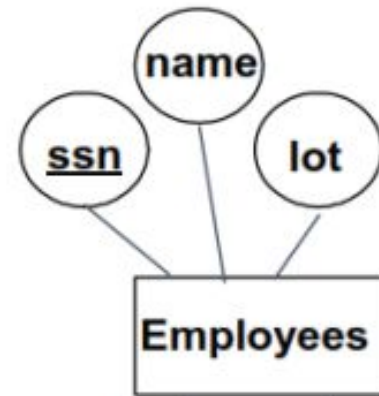
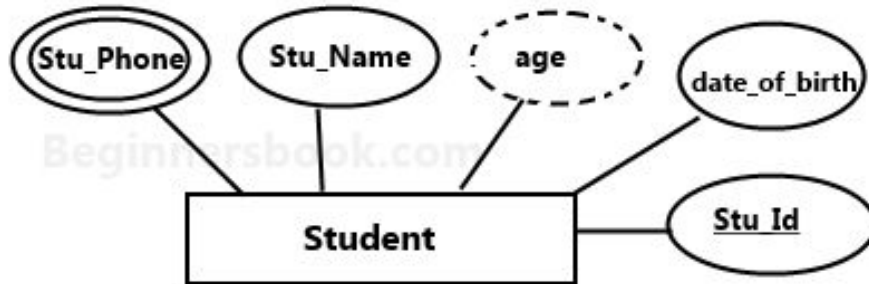


Multivalued attributes

ER diagrams: Key

Each entity set has a key:

a set of attributes uniquely identifying an entity



Entity Types and Entity Sets

Entity Types

- Describes a groups of entities that have similar characteristics/properties: For example, a university has thousands of students that share the same attributes, but each entity has its own value(s) for each attribute.
- An entity type defines a collection (or set) of entities that have the same attributes.
- Each entity type in the database is described by its name and attributes.

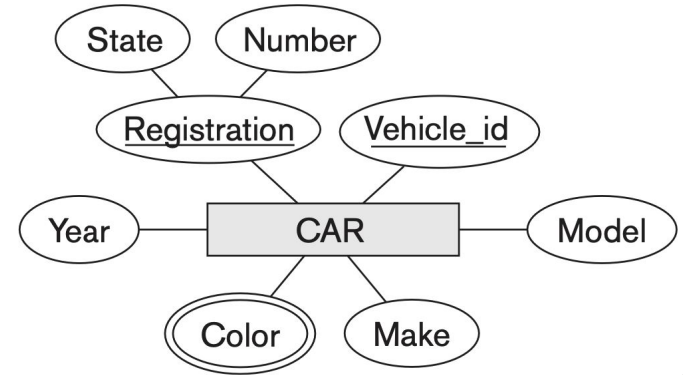
Entity Sets

- The collection of all entities of a particular entity type;

Entity keys

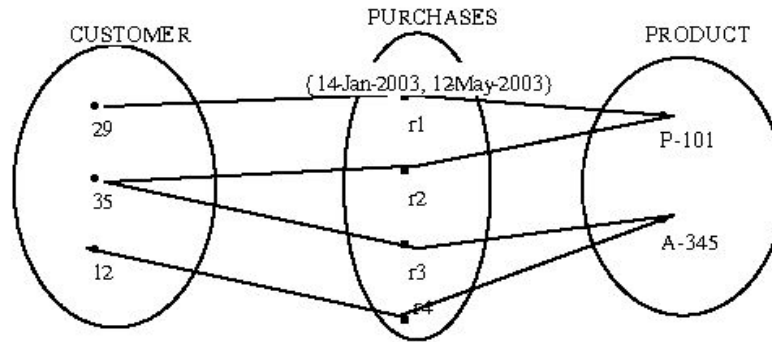
An attribute that uniquely defines an entity in an entity set:

- **Super key:** A set of attributes (one or more) that together define an entity in an entity set
- **Candidate key:** A minimal super key
 - An entity set may have more than one candidate key
- **Primary key:** A candidate key chosen by the database designer to uniquely identify the entity set



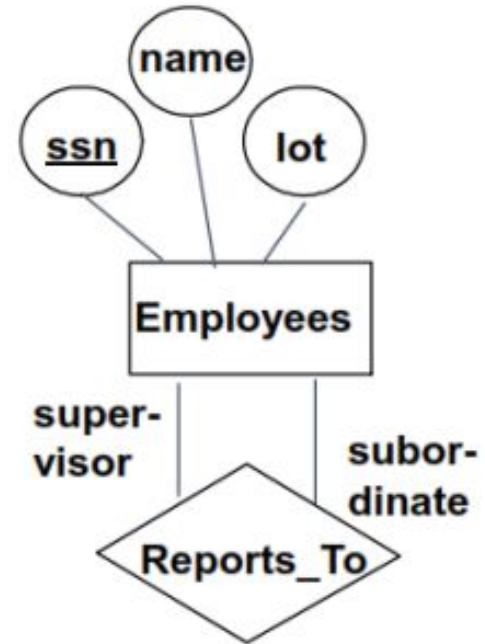
ER diagrams: Relationships

- Relationship is specified whenever an attribute of one entity type refers to another entity type.
- Relationship Types/Sets/Instances
 - A **relationship type** R among n entity types E_1, E_2, \dots, E_n defines a set of associations - or a **relationship set** - among entities from these entity types. Both are referred with R .
 - Mathematically, the relationship set R is a set of relationship instances r_i , where each r_i associates n individual entities (e_1, e_2, \dots, e_n) , and each entity e_j in r_i is a member of entity set E_j , $1 \leq j \leq n$.
 - Each entity type E_1, E_2, \dots, E_n participates in the relationship type R
 - Each individual entity e_1, e_2, \dots, e_n participates in the relationship instance $r_i = (e_1, e_2, \dots, e_n)$.



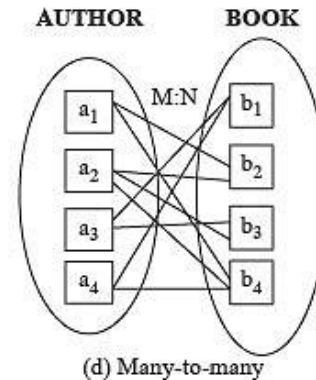
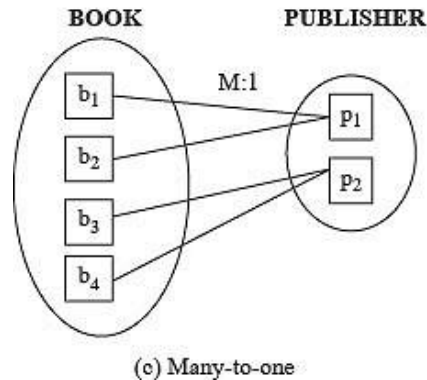
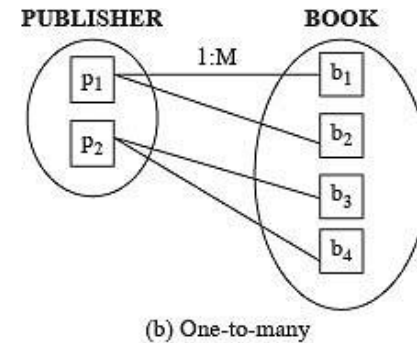
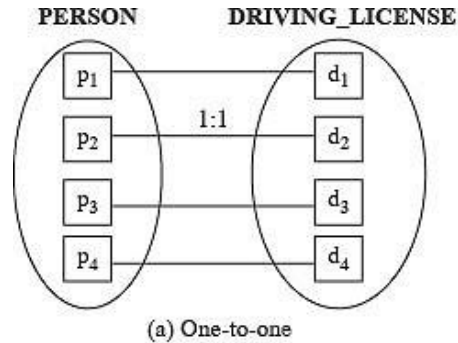
ER diagrams: Relationships

- **Degree of a Relationship Type.** The degree of a relationship type is the number of participating entity types.
- The function that an entity plays in a relationship is called its role.
 - Roles are normally explicit and not specified. They are useful when a clarification is needed.
 - An entity could participate in different relationships, or in different “roles” in same relationship
 - **Recursive relationships:** If a same entity participates more than once
- Relationships can also have attributes;



Cardinality ratios

Express the number of entities to which another entity can be associated via a relationship



Different notations to represent cardinality

one-to-one (1:1)



one-to-many (1:N)



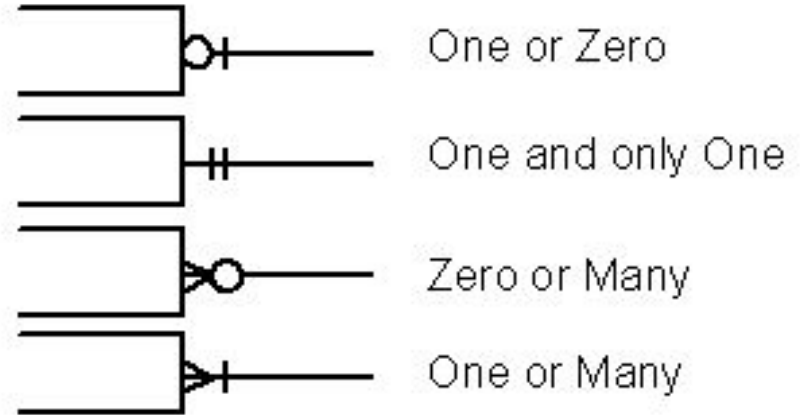
many-to-one (N:1)



many-to-many (M:N)

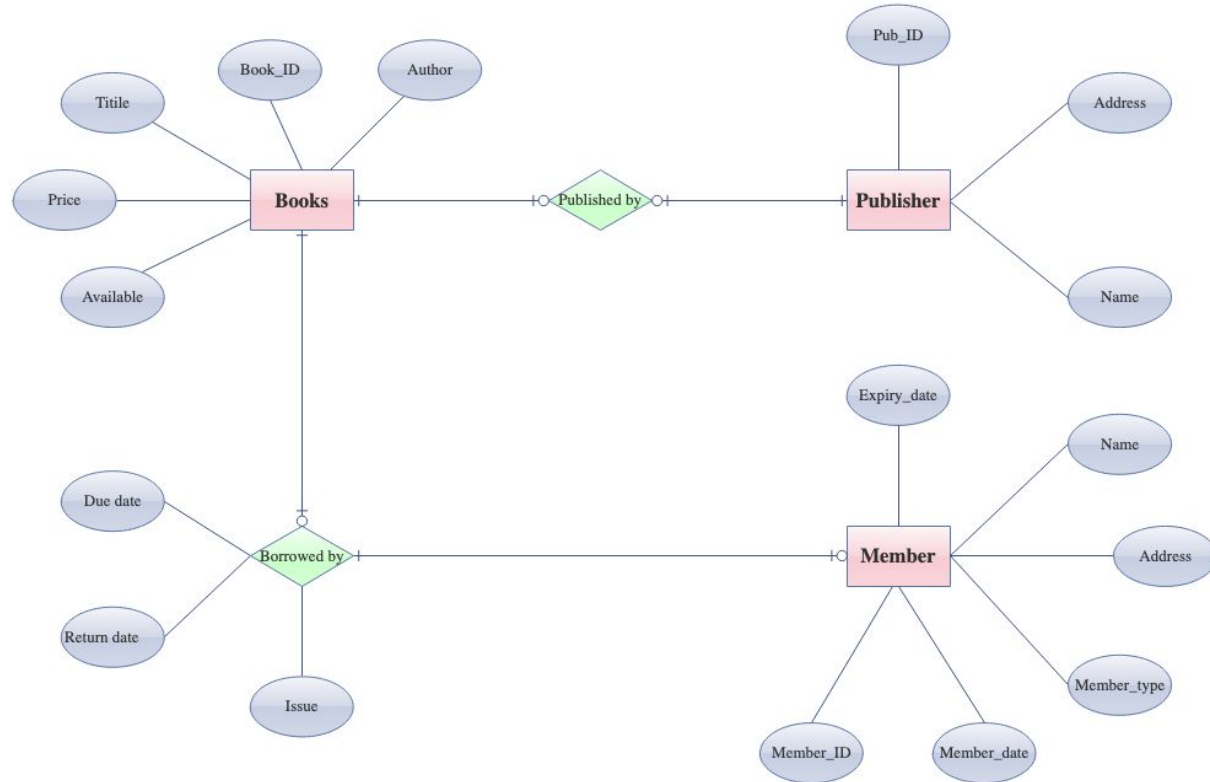


Summary of Crow's Foot Notation



Example of a ER diagram for a Library Management System

Library Management system



Participation Constraints and Existence Dependencies

Participation CONSTRAINTS and CARDINALITIES must be ENFORCED when IMPLEMENTING the database

The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.

- specifies the minimum number of relationship instances that each entity can participate - minimum cardinality constraint.

Total Participation

- Each entity is involved in the relationship: represented by double or thicker lines

Partial participation

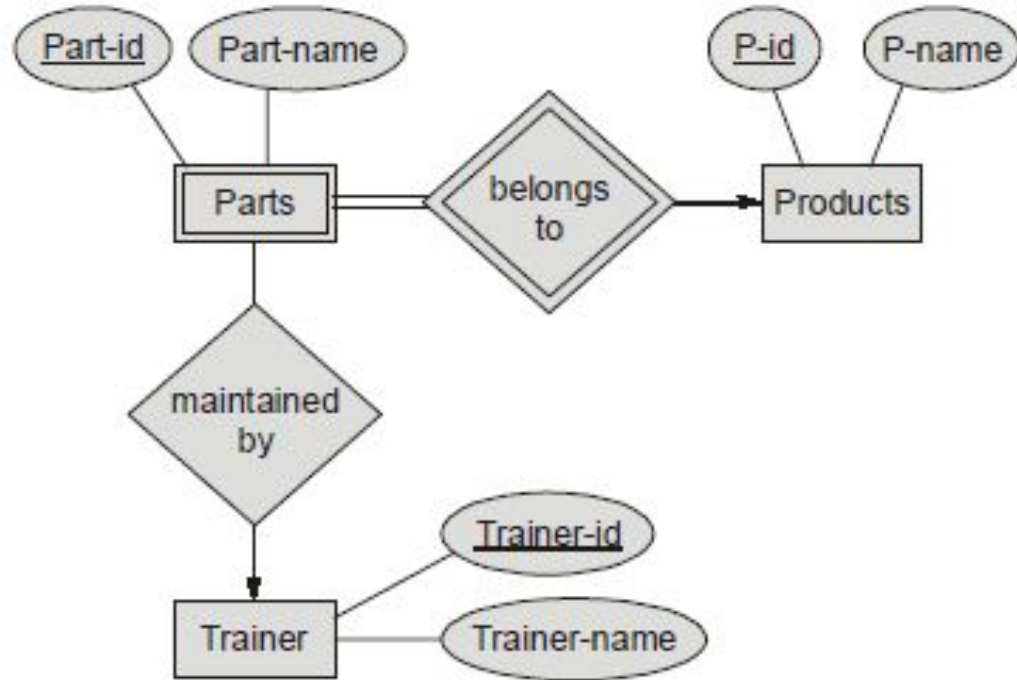
- Not all entities are involved in the relationship: represented by single lines

Strong and Weak Entity Types

- Entity types that do have a key attribute - are called strong entity types.
- Entity types that do not have key attributes of their own are called weak entity types.
- Weak entities are identified with one of their attribute values in combination with another entity type (owner entity type).
- Weak entity type always has a total participation constraint with respect to its identifying relationship because a weak entity cannot be identified without an owner entity.
- **Notation:** A weak entity type and its identifying relationship are surrounding their boxes and diamonds with double lines

Weak Relationships

- ❖ Weak relationship is the identifying relationship of a weak entity type:
 - connection that exist between a weak entity type and its owner
- ❖ A weak relationship is indicated by a doubly - outlined diamond



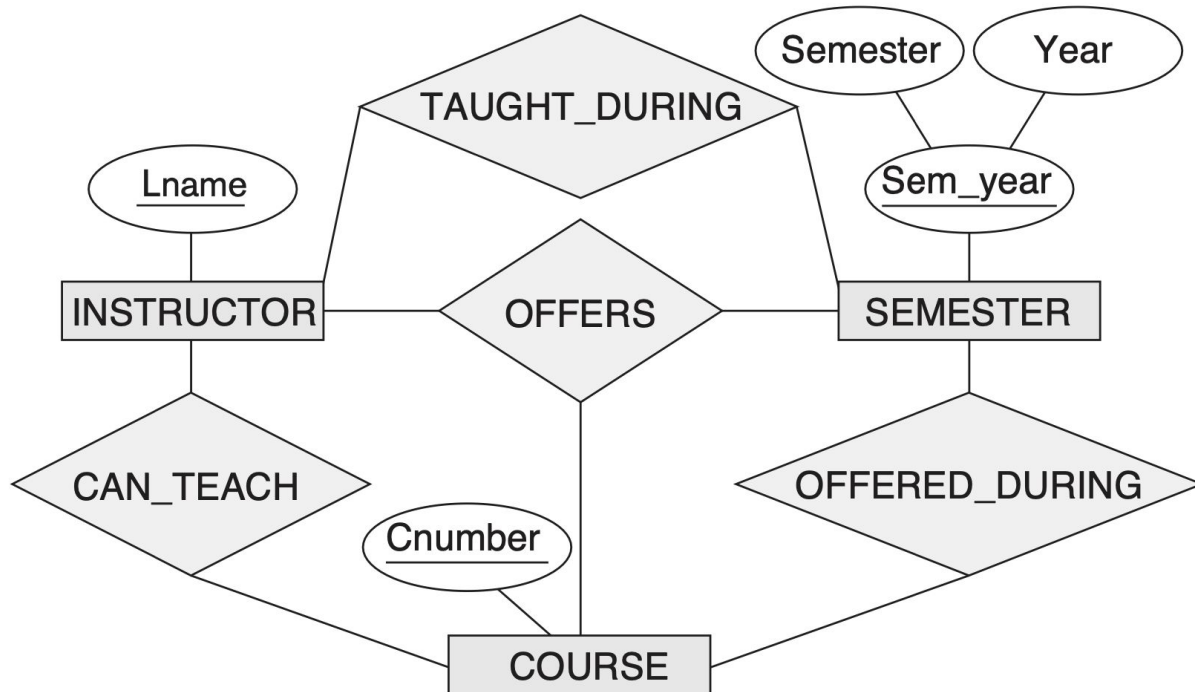
General Design Choices

- ❖ It is occasionally difficult to decide whether a particular concept should be modeled as an entity type, an attribute, or a relationship type.
- ❖ The schema design process is an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached.
- ❖ General Recommendations:
 - It is often the case that a pair of such attributes that are inverses of one another are refined into a binary relationship.
 - Once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to avoid duplication and redundancy.
 - An attribute that exists in several entity types may be elevated or promoted to an independent entity type.

- ❖ ER design is subjective;
- ❖ There are often many ways to model a given scenario

ADVANCED TOPICS

Binary and (or Higher-Degree) Relationships



Binary and (or Higher-Degree) Relationships (2)

- ❖ It is often tricky to decide whether a particular relationship should be represented as a relationship type of degree n or should be broken down into several relationship types of smaller degrees.

The designer must base this decision on the semantics or meaning of the particular situation being represented. The typical solution is to include the ternary relationship plus one or more of the binary relationships, if they represent different meanings and if all are needed by the application















Binary and (or Higher-Degree) Relationships (3)

Three binary relationships cannot replace a ternary relationship.

However, they may do so under certain *additional constraints*. (E.g. if the CAN_TEACH relationship is 1:1, then the ternary relationship OFFERS can be left out because it can be inferred from the three binary relationships CAN_TEACH, TAUGHT_DURING, and OFFERED_DURING).

The schema designer must analyze the meaning of each specific situation to decide which of the binary and ternary relationship types are needed.

Summary of the notation for ER diagrams

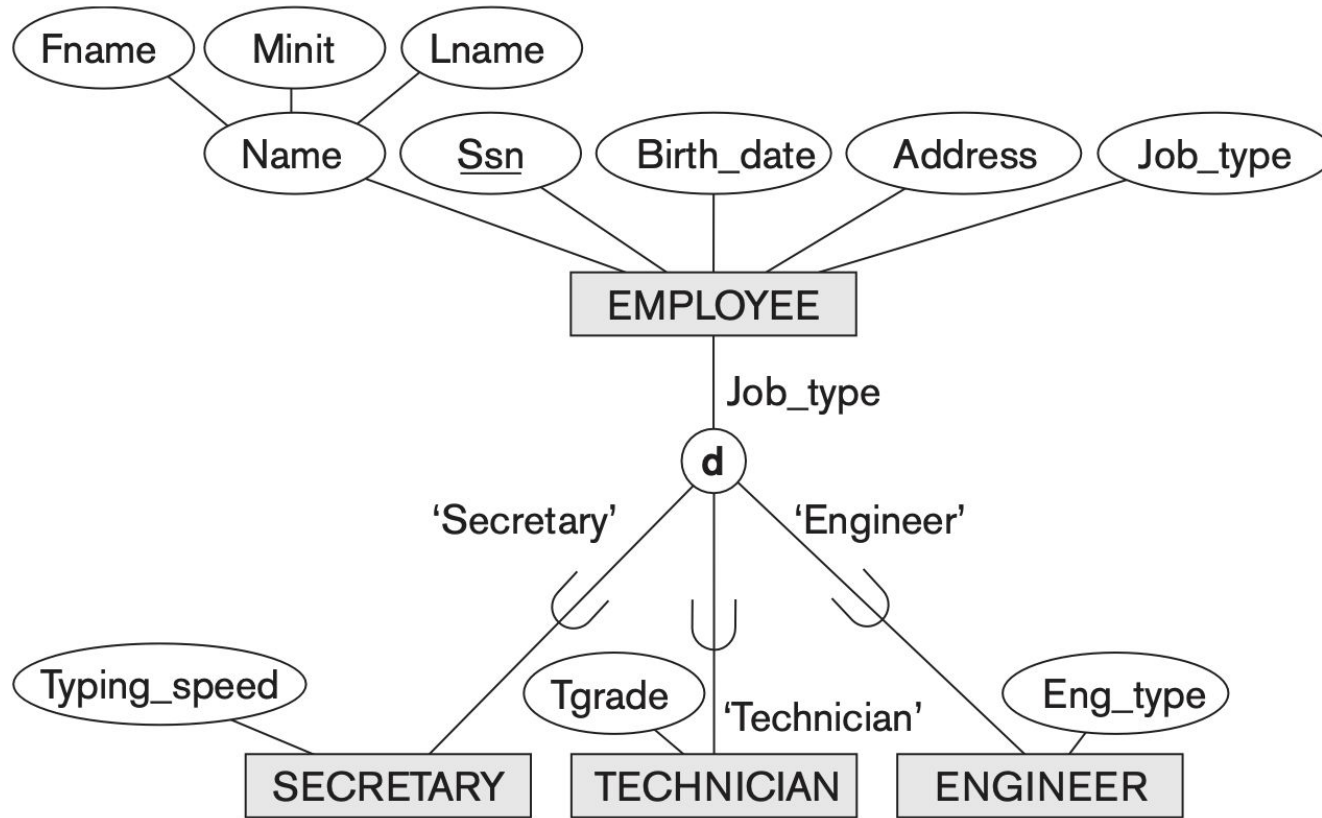
	Entity		Attribute
	Weak Entity		Key Attribute
	Associative Entity		Key Attribute
	Relationship		Key Attribute
	Identifying Relationship		Derived Attribute
	Mandatory Relationship		Optional Relationship
	Partial Participation		Total Participation

Enhanced Entity–Relationship (EER) Model

- New complex requirements for complex applications
- Development of additional semantic data modeling concepts incorporated into conceptual data models
- **Inheritance Type:** Specialization and Generalization
 - **Subclass and Superclass**
 - Object Oriented programming (OOP) style
- **Superclass/Subclass relationship:**
 - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass.
 - Entity that is a member of a subclass inherits all the attributes of the entity as a member of the superclass.

Specialization and Generalization

- Specialization:
 - the process of defining a set of subclasses of an entity type
 - superclass of the specialization
- Generalization:
 - the process of defining a generalized entity type from a set of entity types that share similar characteristics
- Constraints on Specialization and Generalization
 - multiple inheritance
 - hierarchies (single inheritance)
 - **Predicate-defined subclasses:** the entities that will become members of each subclass are determined by a condition in the value of some attribute in the superclass.



EER diagram notation for an attribute-defined specialization on Job_type.

More Constraints for Specialization and Generalization

- ❖ **User-defined subclass:** do not have a condition (attribute) for determining membership in a subclass. Membership in such a subclass is *specified individually for each entity by the user*, not by any condition that may be evaluated automatically.
- ❖ **Disjointness constraint:** specifies that the subclasses of the specialization must be disjoint sets.
 - an entity can be a member of at most one of the subclasses of the specialization.
- ❖ If the subclasses are not constrained to be disjoint, their sets of entities may be overlapping

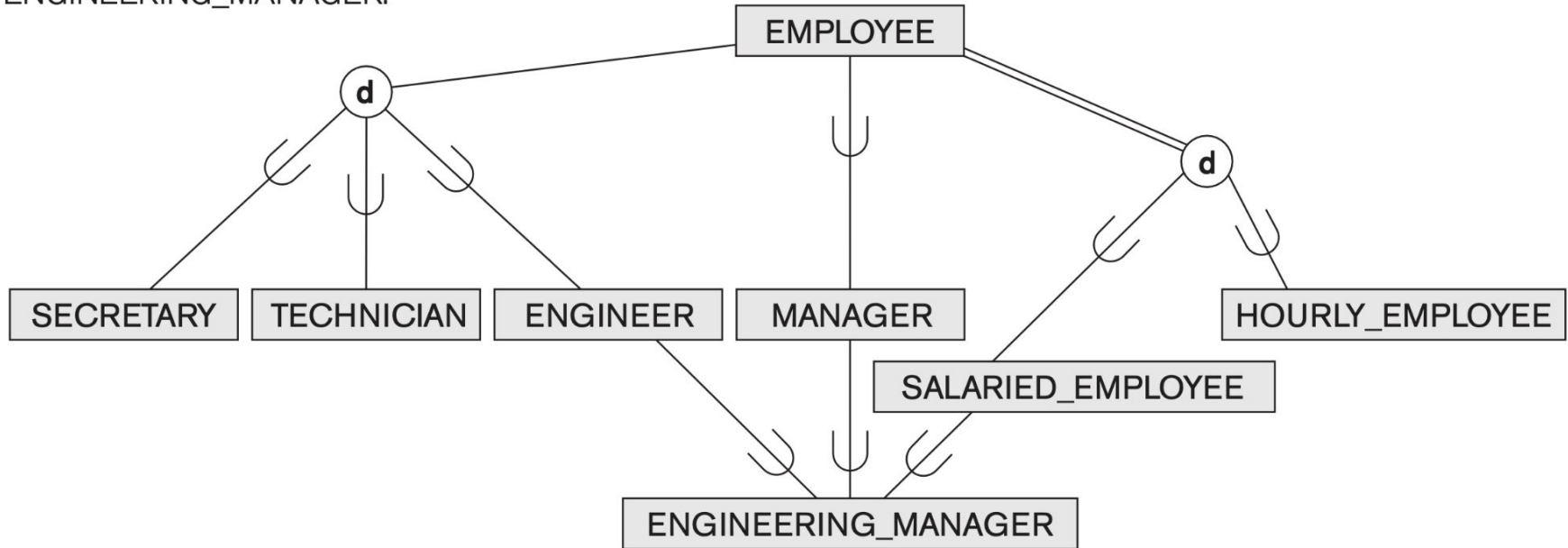
More on Constraints for Specialization and Generalization

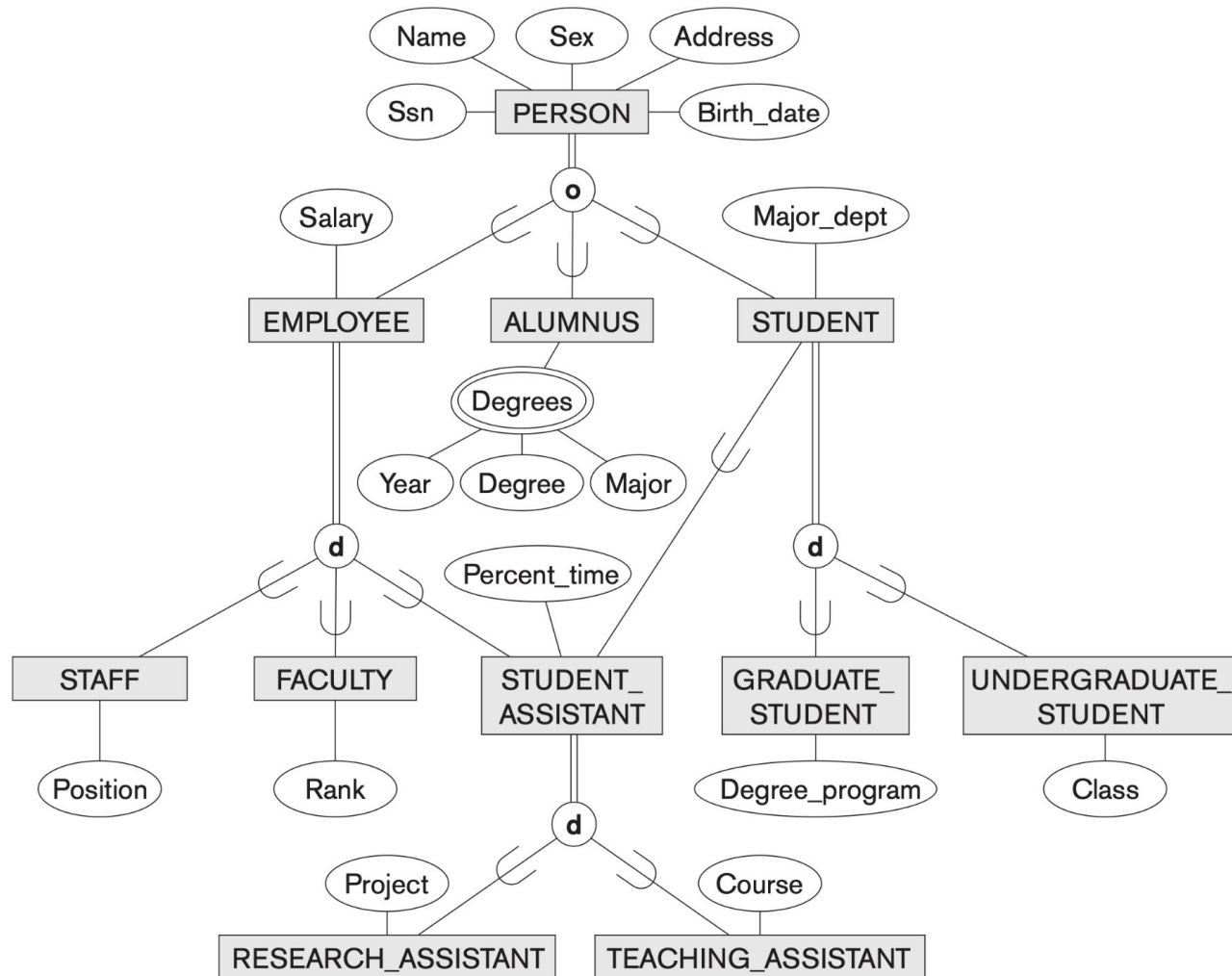
- ❖ **Completeness constraint:**
 - Specialization can be total or partial.
- ❖ **Total specialization** constraint specifies that every entity in the superclass must be a member of at least one subclass in the specialization.
- ❖ **Partial specialization:** an entity might not belong to any of the subclasses.
- ❖ The disjointness and completeness constraints are independent. We have the following four possible constraints on a specialization:
 - Disjoint, total ■ Disjoint, partial ■ Overlapping, total ■ Overlapping, partial

More on Constraints for Specialization and Generalization

- ❖ Superclass that was identified through the generalization process usually is total, because the superclass is derived from the subclasses and hence contains only the entities that are in the subclasses.
- ❖ Certain insertion and deletion rules apply to specialization (and generalization) as a consequence of the constraints specified earlier:
 - Deleting an entity from a superclass implies that it is automatically deleted from all the subclasses to which it belongs.
 - Inserting an entity in a superclass implies that the entity is mandatorily inserted in all predicate-defined (or attribute-defined) subclasses for which the entity satisfies the defining predicate.
 - Inserting an entity in a superclass of a total specialization implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization.

A specialization lattice with shared subclass
ENGINEERING_MANAGER.





Bottom-up conceptual synthesis vs. Top-down conceptual refinement

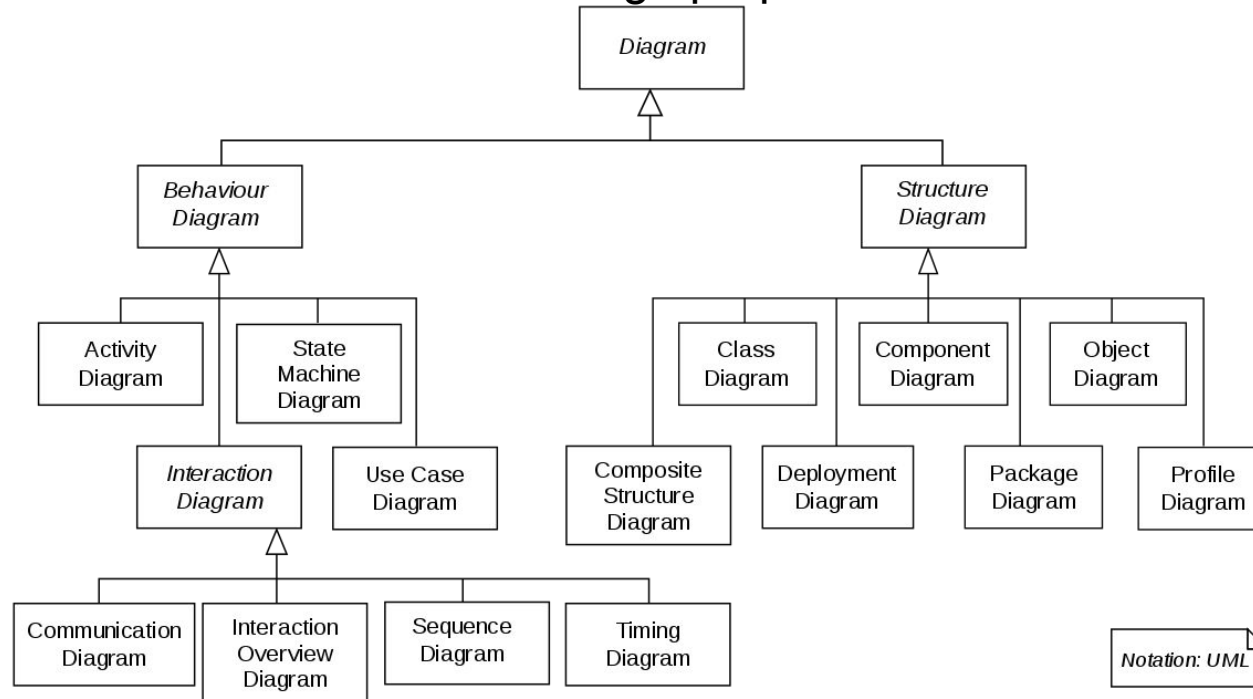
Differences between the specialization and generalization processes:
= they are used differently to refine conceptual schemas during conceptual database design.

- Top-down conceptual refinement: In the specialization process, the database designers typically start with an entity type and then define subclasses of the entity type by successive specialization;
- Bottom-up conceptual synthesis: the process involves generalization of multiple entities in a single “common” entity
- Union type is a construct to represent collection of entities from different entity types that do not share characteristics (distinct entity types)

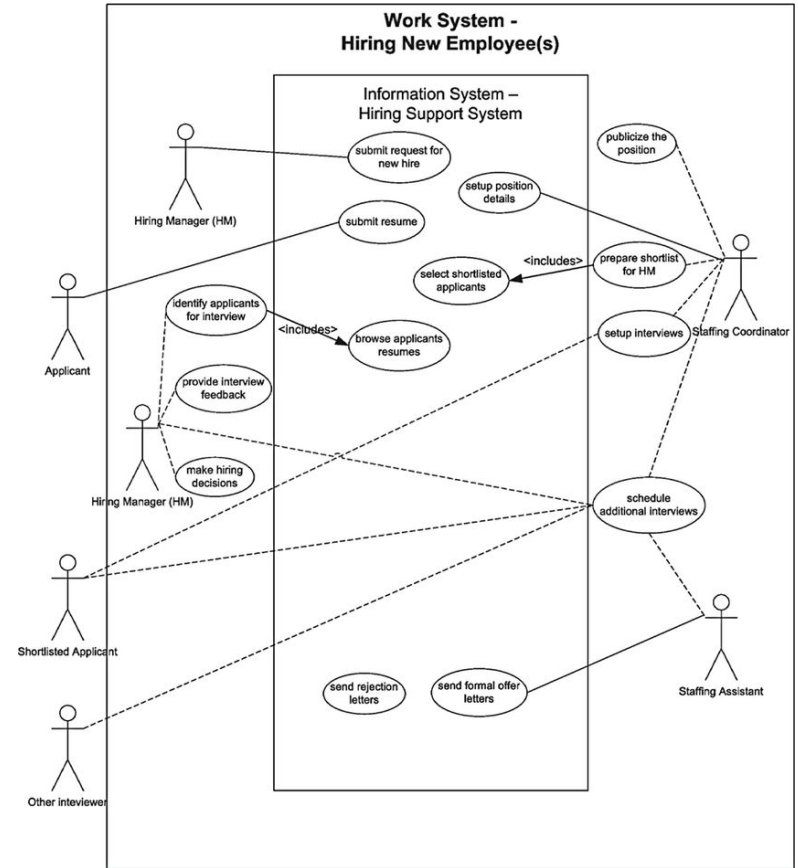
UML - Unified Modeling Language

UML - Unified Modeling Language

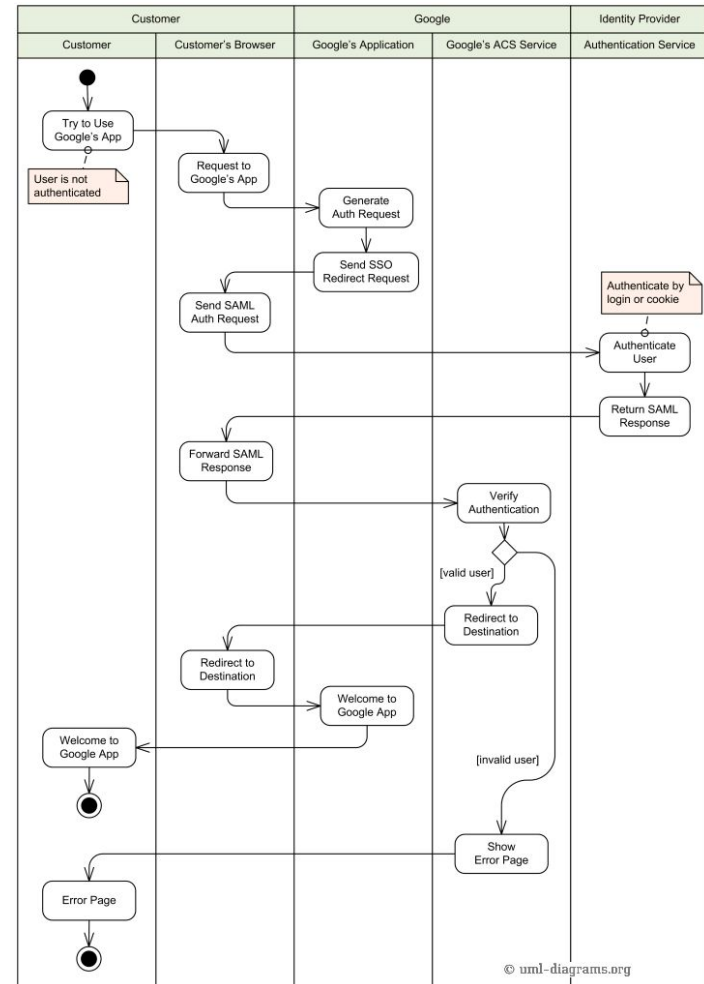
- UML is being used extensively in software design and has many types of diagrams for various software design purposes.



Use Case Diagram



Activity Diagram

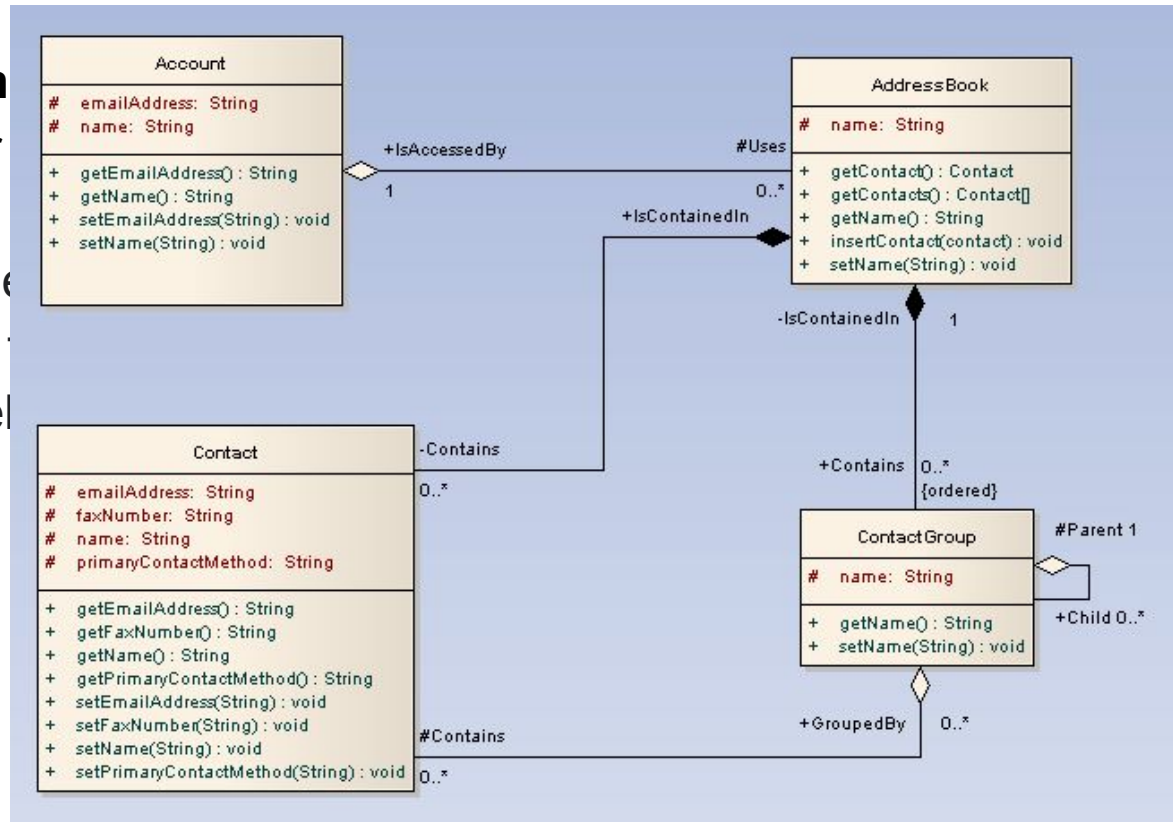


Class Diagrams

- **Unified Modeling Language (UML) notation** for class diagrams has been used as a standard for conceptual object modeling.
- Class diagrams are used to model the structure of an application, and for detailed modeling and translating models into code. Class diagrams can also be used for data modeling.

Class Diagrams

- **Unified Modeling Language** is used as a standard for modeling software systems
- Class diagrams are used for detailed modeling and can also be used for data modeling



UML vs. ER Diagram

UML Motivations:

- UML helps to develop efficient, effective and correct Object-Oriented designs
- UML is used for clear communication between project stakeholders

Entity-Relationship diagrams are not UML diagrams

UML: Object Oriented Design

ER: conceptual data modeling

UML class diagrams are similar to ER, but not the same!

ER and UML

When should UML be used? When should ER be used?

Tool support:

- ER models can be mapped to relational schema
- From UML a code skeleton can be generated in several languages

ER and UML

When should UML be used? When should ER be used?

Different modeling languages (Entity-Relation, Unified Modeling Language, and others) are simply notations for communicating a design to stakeholders. Communicating a design is technical communication, and one of the principles of good technical communication is to communicate the information clearly and concisely. Choosing a modeling notation that is understood by your audience and can communicate the desired information clearly is the first step to achieve this principle.

Mapping between UML and ER: Basic concepts

In UML class diagrams, a **class** is equivalent to an **entity type** in ER.

- The top section gives the **class name** (equivalent to entity type name in ER);
- The middle section includes the **attributes**;
- The last section includes **operations** that can be applied to individual objects (similar to individual entities in an entity set) of the class. Operations are *not* specified in ER diagrams.

○

BankAccount
owner : String balance : Dollars = 0
deposit (amount : Dollars) withdrawal (amount : Dollars)

Mapping between UML and ER: Basic concepts (2)

- Relationship types (ER) are equivalent to **associations** in UML
- Relationship instances are equivalent to **links**.
- A **binary association** (binary relationship type) is represented as a line connecting the participating classes (entity types), and may optionally have a name. A relationship attribute, called a **link attribute**, is placed in a box that is connected to the association's line by a dashed line.

Mapping between UML and ER: Basic concepts (3)

- The **(min, max) notation to specify relationship constraints** are equivalent to **multiplicities** in UML terminology.
- Multiplicities are specified in the form min..max, and an asterisk (*) indicates no maximum limit on participation
- **Recursive relationship type** is equivalent to **reflexive association** in UML
- **Aggregation**: a relationship between a whole object and its component parts (different notation is used)
- **Association (relationship) names are optional** in UML, and relationship attributes are displayed in a box attached with a dashed line

Mapping between UML and ER: Basic concepts (4)

- Weak entities (in ER) are modeled using the construct qualified association (UML)
- In UML terminology, the partial key attribute is called **discriminator**, because its value distinguishes the objects associated with (related to) the same entity

Knowledge Representation and Ontology

- ❖ Conceptual data modeling
- ❖ Artificial Intelligence (AI)
- ❖ KR techniques: to develop concepts for accurately modeling some domain of knowledge. **Output:** An ontology
- ❖ Ontology: describes the concepts of the domain and how these concepts are interrelated through some common vocabulary.
- ❖ The main difference between an ontology a database schema, is that the schema is usually limited to describing a small subset of the world to store and manage data. An ontology is considered to be general in that it attempts to describe a part of reality or a domain of interest.

