Conceptual Design Using the Entity-Relationship (ER) Model
Overview of Database Design

• Conceptual design, using ER model
  • What are the entities and relationships in the enterprise?
  • What information about these entities and relationships should we store in the database?
  • What are the integrity constraints or business rules that hold?

We will then map the ER diagram into a relational (SQL) schema.
Overview of Database Design (cont.)

- **Schema Refinement, or Normalization**
  Check relational schema for redundancies and related anomalies.

- **Physical Database Design and Tuning**
  Consider typical workloads and further refine the database design, with appropriate indexes, etc.
ER Model Basics

• **Entity:** Real-world object distinguishable from other objects.
  
  An entity is described using a set of attributes.

• **Entity Set:** A collection of similar entities (e.g., all employees).
  
  • All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies!)
  
  • Each entity set has a key.
  
  • Each attribute has a domain.
  
  • Can map entity set to a relation easily.
• **Relationship:** Association among 2 or more entities. E.g., Jane works in Pharmacy Department.

• **Relationship Set:** Collection of similar relationships.

  An n-ary relationship set R relates n entity sets $E_1 \ldots E_n$;
  each relationship in R involves entities $e_1, \ldots, e_n$.

  Same entity set could participate in different relationship sets, or in different “roles” in the same relationship set.
ER Model Basics (Cont.)

Relationship sets can also have descriptive attributes (e.g., the since attribute of Works_In).
Key Constraints

- Consider Works_In: An employee can work in many departments; a dept can have many employees.
- Consider Manages: In contrast, each dept has at most one manager, according to the key constraint for Departments on Manages.
Participation Constraints

Does every department have a manager?

If so, this is a participation constraint: the participation of Departments in Manages is said to be total (vs. partial)
A weak entity can be identified uniquely only by considering the primary key of its owner entity.

- Owner entity set and weak entity set must participate in a binary relationship set called the identifying relationship set.
- Weak entity set must have a key constraint and total participation in this identifying relationship set.
ISA (‘is a’) Hierarchies

- If we declare A ISA B, every A entity is also considered to be a B entity.
- As in C++, or other languages, A inherits all B’s attributes.

**Overlap constraints:** Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (Allowed/Disallowed)

**Covering constraints:** Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (Yes/No)

**Reasons for using ISA:**
- To add attributes specific to a subset of entities
- To identify a subset of entities that participate in a relationship
Aggregation

Used when we have to model a relationship set involving (entity sets and) a relationship set.

Aggregation allows us to treat a relationship set as an entity set in another relationship set.

Aggregation vs. ternary relationship:
- Monitors is a distinct relationship, with a descriptive attribute.
- Also, could say that each sponsorship is monitored by at most one employee.
Conceptual Design Using the ER Model

• **Design choices**
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Aggregation?

• **Constraints in the ER Model**
  - Much data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams: we write them precisely in plain English as an annotation to the ER diagram, so we can later remember and capture them in the SQL schema.
Entity vs. Attribute

Should *address* be an attribute of Employees or an entity set (connected to Employees by a relationship)? Really depends on intended use of addresses in our database, and on data semantics:

- If the address structure (city, street, etc.) is important, so that, say, we can retrieve employees by city, address must be modeled as an entity set (attribute values are “atomic” in our version of the ER Model).
- If we have several addresses per employee, address must be an entity set (attributes cannot be set-valued).
Works_In2 does not allow an employee to work in a department for two or more periods.
Entity vs. Relationship

- Manages2 is OK if a manager gets a separate discretionary budget for each dept.
- Manages2 is not OK if a manager gets a discretionary budget that covers all managed depts.: redundancy of dbudget, (same value stored for each dept. managed by a manager); misleading design (suggests dbudget tied to managed dept.)
If each policy is owned by just 1 employee, a key constraint on Policies would mean policy can only cover 1 dependent!

What are the additional constraints in the 2nd diagram?
Summary of Conceptual Design

- Conceptual design follows requirement analysis, to yield a high-level description of data to be stored
- ER model popular for conceptual design: ER constructs are expressive, close to how we think about applications
- Basic ER constructs: entities, relationships, and attributes (of entities and relationships)
- Some additional ER constructs: weak entities, ISA hierarchies, and aggregation

Note: There are many ER model variants. So that we know what you mean, please stick strictly to the version we covered in class for your homework and projects.
Summary of ER (Contd.)

• ER design is subjective and often there are many (proper) ways to model a given scenario. Common choices include:
  
  Entity vs. attribute, entity vs. relationship, binary vs. n-ary relationship (n>2), whether or not to use ISA hierarchies, and whether or not to use aggregation.

• Ensuring good database design: resulting relational schema should be analyzed and refined further. More on this later...