Entity Relationship Model (ERM)

Kathleen Durant
CS3200
Lecture 2
What is the goal of Modeling?

- Derive a logical description of our data.
- Understand the various ways in which the data is used.
- Identify the important or central data.
- Make decisions about the relationships between data and how to decompose a design.
- Organize the data to facilitate its uses.
- Be reasonably efficient.
- Allow for efficiency of implementation.
Why use a model?

- The data is probably reasonably complex.
- There are several different sorts of accesses.
- The data and design will evolve over time:
  - You want a history of this evolution.
  - You want to be able to make changes without constantly dumping and loading your data.
  - Good mental structuring will lead to good physical structure.
- The act of organizing will make people think of things that got dropped on the floor.
How do you develop a model?

Steps to model development

- Identify the entities.
- Determine all significant interactions.
- Analyze the nature of the interactions.
Entity relationship model (ERM)

- Maps nicely into a relational data model.
- Provides a set of terminology and a graphical display of the data.
- Fairly simple to understand.

Alternatives
- Process and data flow analysis.
- Seat of the pants methodology.
  - That’s sloppy thought
  - Leads to a sloppy database with excessive redundancy, poor performance, and inability to get the job done.
The ER Model: Entity

- **Entities:** All the world is a set of *things*.
  - Represented by a rectangle.
  - They have their attributes and relationships.
  - An entity is one object, it is described via its attributes.

- **Entity Set:** A collection of similar entities, e.g., all movies.
  - Typically, all entities in an entity set have the same set of attributes.
    - (Exception: ISA hierarchies violate this condition.)
  - Each entity set has a key.
The ER Model: Attributes

• Attributes:
  • Describe the entities.
  • There are many different types of attributes
  • **Represented by an oval. Attached to entities with a line.**
  • Each attribute has a domain.
    • A set of potential values.
• Analogies:
  • Fields in a record
  • Elements of a data structure or class object
Types of Attribute

• **Simple**: indivisible (like a native type). Examples?
• **Composite**: decomposable or structured. Examples?
• **Single-valued**: only one per entity. Examples?
• **Multi-valued**: zero or more per entity. Examples?
  • Represented by a double oval
• **Domain of an attribute**: it’s possible values
• **Key**: subset of attributes that uniquely identifies an entity (candidate key)
Example: from IMDB

- Movie
  - Director
  - Title
  - Release Date

- Actor
  - Birthdate
  - Name
  - Photo
Attribute details

- Null attributes
  - Do not know the value (NA).
  - Attribute does not exist (e.g., children).
- Derived values
  - An attribute that can be computed from other attribute(s)
  - Represented by a dashed oval.
Example: from IMDB

Movie
- Director
- Title
- Release Date

Actor
- Name
- Birthdate
- Age
- Photo
Relationships

• A relationship is a mapping or an association between two or more entities.
  • Relationship represented as a line connecting two entities or entity sets.
• A Relationship set: a collection of similar relationships, mapping between 2 entity sets.
  • Represented by diamonds

Key design issue: distinguishing between entities and relationships -- seems obvious but isn’t always.
Degree of a Relationship

• Degree of relationship identifies the **number of entities that participate in the relationship.**
  • Binary relationships are most common.
  • Higher degree relationships can be modeled as a set of binary relationships.
Example: Relationships

Relationships can have attributes too.
Relation Constraints: Cardinality

- The logical structure of the data may impose constraints – ERM allows you to represent these constraints.
- Cardinality: defines the relationship between the entities in terms of #’s.
- 1 to 1: represented as 2 arrows pointing into the relationship.
- 1 to many: one arrow emanating from the entity set with the cardinality of many (or pointing to the 1 cardinality).
- Many to many.

Diagram:

- Movie
  - is
  - in
  - Actor
- Movie
  - is
  - in
  - Actor
- Movie
  - is
  - in
  - Actor
Describe the wacky Movie world

Each actor can be in 1 movie and each movie has 1 actor.

Each actor can only be in 1 movie but a movie can have 0, 1, or n actors.

A movie can have 0, 1, or n actors and an actor can appear in 0, 1, or n movies.
Constraints: Existence

• Entity X is *existent dependent* on Entity Y if X can only exist if Y exists.
  • Has implications for deletion:
    • Deleting Y must force delete of X as well.

• In this example:
  • X is the *subordinate* entity
  • Y is the *dominant* entity
Constraints: Participation

- If every entity in an entity set E must be part of a relationship R, then E participates totally (as opposed to partially).
- Total participation often indicates existence dependencies.
- Total participation represented by a double line.
Existence: Example

• Assume actors only exist in the IMDB database if they have been in at least one movie
• If you delete a movie from IMDB
  • Must also delete actors that only appeared in that one movie
Types of Keys

- **Superkey**: an attribute or set of attributes that uniquely identifies an entity—there can be many of these
  - **Candidate key**: a superkey such that no proper subset of its attributes is also a superkey (minimal superkey—has no unnecessary attributes)
- **Primary key**: the candidate key chosen to be used for identifying entities and accessing records. Unless otherwise noted "key" means "primary key"
  - Represented with an underline
  - Used for physical clustering of data
- **Alternate key**: a candidate key not used for primary key
- **Secondary key**: attribute or set of attributes commonly used for accessing records, but not necessarily unique
- **Composite key**: a key requiring more than one attribute
Example: Keys

- **Actor ID**
- **Birthdate**
- **Age**
- **Role**
- **Name**
- **Photo**
- **Director**
- **Title**
- **Release Date**

Relationships:
- Movie is in Actor
- Actor ID is in Actor
- Role is in Actor
- Name is in Actor
- Photo is in Actor
- Director is in Movie
- Title is in Movie
- Release Date is in Movie
Weak vs. Strong entity sets

- An entity set without a primary key is called a strong entity set
  - Represented by a double rectangle
  - Corresponding relationship represented with a double diamond
- A discriminator (*partial key*) distinguishes among elements of a weak entity set.
- An entity set with a primary key is called a strong entity set
- Members of the strong entity set are dominant; members of the weak entity set are subordinate
Example: Weak entity set

Diagram:
- Scene
- Part of
- Movie
- Actor
- Actor ID
- Birthdate
- Age
- Number
- Role
- Name
- Photo
- Director
- Title
- Release Date

Relationships:
- Scene is part of Movie
- Movie contains Actor
- Actor is in Role
- Role is in Movie
- Actor has Name
- Actor has Photo
- Actor has Birthdate
- Actor has Age
Extended features

• Specialization
  • Designating subgroups within an entity set (think subclasses or sub-typing).
  • A single entity set can have multiple specializations.
  • Represented using a triangular ISA designator.

• Generalization
  • Inverse of specialization.
  • Aggregates similar entity sets
  • Represented using a triangular ISA designator.

• Generalized entity set or the *basis* of a specialization is called a *higher level entity set.* (The others are *lower level entity sets.*)
  • Familiar if you have used objects:
  • Superclass/subclass.
  • Inheritance.
  • If the entire schema has only single inheritance, then the design is a hierarchy.
  • If the schema has multiple inheritance, it is a lattice.
Generalization

ISA function defines a Hierarchy

Specialization

Movie

Higher Entity Set

Action

Drama

Comedy

Lower Entity Set
Generalization constraints

• **Membership**
  - Condition-defined: membership in the generalization or (specialization) is based on a **predicate**.
  - User-defined: membership is defined manually.

• **Cardinality**
  - Disjoint: each entity belongs to a single lower level entity set.
    - Single genre for a movie
  - Overlapping: entities may belong to multiple lower level entity sets.
    - Movie classified using multiple genres

• **Completeness**
  - Total: every higher level entity must belong to a lower level entity set.
    - All movies have at least one genre.
  - Partial: higher level entities may or may not belong to lower level entity sets.
    - Some movies do not fall into any movie genre.
Aggregation

• When relationship sets have their own relationships
• Provides a method to build up complicated entities
• Allows us to treat a relationship set as an entity set
  • Represented as a box around the relationship
Summary of Conceptual design

- Conceptual design follows requirements analysis
  - Yields a high-level description of data to be stored
  - ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications
  - Basic constructs: entities, relationships, and attributes (of entities and relationships)
  - Some additional constructs: weak entities, ISA hierarchies, and aggregation
    - Note: There are many variations on ER model
Summary of ER

- ER design is subjective. There are often many ways to model a given scenario. Analyzing alternatives can be tricky, especially for a large enterprise.

- Common choices include:
  - Entity vs. attribute
    - Key for the entity / to store or discard an attribute
  - Entity vs. relationship
  - Binary or n-ary relationship
  - Use of ISA hierarchies
  - Use of aggregation
Concepts: Part 1

- Entity: a thing (abstract or concrete).
- Relationship: mapping among entities.
- Enterprise Schema: overall logical schema of a database.

- Entity set: a collection of entities all of which have the same attributes.
- Relationship set: the mapping between entity sets
- Extension: the individual entities that comprise an entity set.

- Attribute(s): properties that describe an entity or relationship.
  - Domain (value set): permitted values of an attribute.
  - Simple attribute: indivisible type.
  - Composite attribute: attribute may be further broken down into subfields.
  - Single-valued attribute: only one entry for the attribute of a specific entity.
  - Multi-valued attribute: may have multiple entries for the attribute, all for a specific entity (e.g., phone numbers: work, home, cell, fax).
  - Derived attribute: attribute whose value can be determined based upon other data (e.g., a database that includes birthdate and age; age can be a derived attribute given birthdate).
  - Base attribute: an attribute from which you derive another attribute.
  - Descriptive Attributes: attributes added to a relationship.
Concepts: Part 2

- Participation: the act of an entity belonging to a relationship
  - Total participation: all entities participate in the relation.
  - Partial participation: not all entities participate in the relation.
- Role: when an entity has a relationship with itself. The role distinguishes how an entity is treated in a relationship.
- Mapping cardinality (1:1, 1:many, many:1, many:many)
- Existence dependencies: requiring an entity to exist if another entity exists.
  - Subordinate entity: the dependent entity in an existence dependency.
  - Dominant entity: the entity on which the subordinate entity exists.
Concepts: Part 3

- **Superkey**: a set of attributes that uniquely identifies an entity.
- **Candidate key**: the minimal set of attributes that forms a superkey.
- **Primary key**: a designated candidate key.

- **Weak entity set**: an entity set without a candidate key.
- **Strong entity set**: an entity set with a candidate key.
- **Discriminator**: a set of attributes that distinguishes between the elements of a weak entity set.
Concepts: Part 4

- **Specialization**: extracting a subclass from an entity set.
- **Generalization**: combining one or more entity sets into a higher level entity.
  - **Disjoint generalization**: an entity belongs to at most one lower level entity set.
  - **Overlapping generalization**: entities may belong to multiple lower level entities.
- **Hierarchy**: each entity set is only the object of one “ISA” relationship.
- **Lattice**: entity sets may belong to multiple “ISA” relationships.
- **Condition-defined constraint**: defines membership in a subclass via a predicate.
- **User-defined constraint**: membership is manually defined.
- **Completeness constraint**: all entities belong to a lower level entity.
- **Total constraint**: all entities belong to lower level entity sets.
- **Partial Constraint**: entities not required to belong to lower level entity set.
- **Aggregation**: grouping part of a schema into a larger unit.
In Class Work

Layout an ERM diagram for a university. The university consists of a number of departments, in particular (Engineering, Humanities, Math, Science). Each department offers several majors. A collection of courses define the minimal collection of courses that satisfy a specific major. Students declare a specific major and take courses towards the completion of that major. Each course is taught by a professor from the appropriate department.