High Level Database Models

Thomas Schwarz, SJ
Contents

• Design Language:
  • Entity Relationship Model (ERM)
  • Unified Modeling Language (UML)
  • (Object Description Language ODL)
E/R Model

- **Entities**: Abstract object
  - Have Attributes
    - Types can be primitive or structures
- **Relationships**
  - Connections between two or more entity sets
E/R Model

• Graphics
  • Entities are represented by rectangles
  • Attributes are represented by ovals
  • Relationships are represented by diamonds
  • Edges connect attributes and relations
E/R Model

- Type of binary E/R relationships between entities:
  - Many-to-one
  - One-to-one
  - Many-to-many
E/R Model

- Example:
  - One president can “run” one studio
  - One studio can only be ‘run” by one president

- The arrow does not guarantee existence, only uniqueness
E/R Model

• Ternary relationships
  • Occasionally, relationships involve more than two entities

  ![Diagram of E/R Model]

  • Contracts involve a studio, a star, and a (set of) movies
  • Each relationship is a triple \((\text{star, movie, studio})\)
E/R Model

- The many-to-one relationship means that for a star and for a movie, there can only be one studio.
- However, a star can have a contract over many movies.
- The studio can contract with several stars for a given movie.
E/R Model

- The arrow notation is limited
  - Studio is only a function of the movie
  - Diagram cannot distinguish between
    - Studio is a function of movie
    - Studio is a function of movie and star
E/R Model

• Roles
  • Entities can appear several times in a relationship
  • Question: Explain the arrow heads or their absence
E/R Model

- Example for a multi-way relationship and an entity set with multiple roles
- Hollywood stars would “belong” to a studio that could lent them out to another studio
E/R Model

- Relationships can also have attributes
- The attribute is functionally dependent on all parties to the relationship
E/R Model

- Some models (UML, ODL) limit relationships to binary
- Move attributes to an entity set
E/R Model

- Multi-way relationships can be modeled through an entity as well.
E/R Model

• Subclasses
  • Some entities are special
    • Use an is-a relationship (a triangle)
Design Principles

• Faithfulness
  • Can be difficult: Is “teaches” between instructors and courses many to many or one to many?

• Avoid Redundancy
  • Example: Add relationship ‘owns’ between movies and studios and add an attribute “studio” to movies.
    • This results in an update anomaly
Design Principles

• Simplicity
  • Avoid introducing more elements than is necessary
    • A studio can own movies, so each studio has a holding
    • Could be represented by this diagram, but entity holdings can also be done away with

  - Keeping it just means more storage space and longer computations
Design Principles

• Smart Selection
  • Not every relationship in the real world is worth-while using
  • Information can be redundant
    • Assume relationships contracts, stars-in, and owns
      • Since a movie has an owning studio, and the owning studio has contracts for each star, we do not need the stars-in relationship
Design Principles

• Picking the right kind of element

• Should studio be an entity set or can we add its attributes to a movie

• Depends on the number of attributes for studio

  • If there is only studio name, we can incorporate it in movies
  
  • If there are more attributes, we probably run into an update anomaly
Constraints in the E/R Model

• Keys
  • Every entity set must have a key
  • There can be more than one key
  • For is-a relationships:
    • Root entity set needs to have all the attributes for a key
Constraints in the E/R Model

- Representing keys: Underline attributes that make up the primary key
Constraints in the E/R Model

- Referential Integrity Constraint
  - E.g. Foreign key constraint
  - Example:
    - Every movie has at most one studio owning it
    - Every movie is owned by a studio
    - Every studio has at most one president
    - Every president has a studio to run
Constraints in the E/R Model

- Use rounded arrows to indicate existence of the foreign entity:
  - Every movie is owned by a studio (existence)
  - But not owned by more than one studio (uniqueness)
Constraints in the E/R Model

- Use rounded arrows to indicate existence of the foreign entity:
  - Every president needs to run a studio
  - Cannot run more than one studio
  - If (s)he stops running a studio, they get deleted from the president table
Constraints in the E/R Model

- Use rounded arrows to indicate existence of the foreign entity:
  - A studio cannot have more than one president
  - But if the president has been fired, the studio still persists
Constraints in the E/R Model

- Use rounded arrows to indicate existence of the foreign entity:
  - A studio does not need to own a movie
  - But it can own more than a single movie
Constraints in the E/R Model

• Degree constraints
  • Limit the number of entities that can be connected to an entity set
    • The same star can only appear in 10 movies
Constraints in the E/R Model

• Degree constraints
  • $\leq 1$ means pointed arrow
  • $= 1$ means curved arrow
Weak Entity Sets

- An entity’s key can be composed of attributes belonging (all or some) to another entity
- Called *weak entity sets*
Weak Entity Sets

• Example:
  • Movie studio has several film crews, given by a number
    • (First unit, second unit, ...)
  • The numbering can be used also by other studios
Weak Entity Sets

- Double stroke indicates a weak entity set
- Crews has key \((\text{number}, \text{studios.name})\)
- Mediated through the “unit-of” relationship
Weak Entity Sets

- Biological species are given by genus and species
  - *Homo neanderthalensis*
- First is genus, then species
- The species has a key: *(species.name, genus.name)*
Weak Entity Sets

- Connecting entity sets used to replace ternary relationships
- Often have no attributes of their own
- Keys are attributes of other entities
Weak Entity Sets

- Contracts have a key made up of stars.name, studio.name, movies.title, movies.year
- Own attribute salary is not a key
Weak Entity Sets

- Key attributes for weak entity sets:
  - Made up of zero or more of its own attributes
  - Key attributes from entity sets that are reached by certain many-to relationships
  - These are called supporting relationships and supporting entity sets, resp.
Weak Entity Sets

- R is a supporting relationship for E to F if the following conditions are true:
  - R binary, many to one or one to one
  - R must have referential integrity:
    - For every E, there must be exactly one F entity in R
    - The attributes in F that supply (parts of the) key for E are also keys for F
Weak Entity Sets

- However, if F itself is weak, then the key attributes for F might be supplied by an entity G, etc.
Weak Entity Sets

- If there are several different supporting relationships:

  - Each relationship is used to supply a copy of the key attributes of F to help form the key of E

  - The relationships can associate an entity \( e \in E \) with different entities \( f_1, f_2 \in F \) and so the parts of the key of E can come from different entities
Weak Entity Sets

- Example

- Each crew is unique
- But to identify a crew, we need data from the supporting relationship
- There needs to be a deterministic process to obtain this data.
Weak Entity Sets

- Example

Values for a crew are obtained from their attributes *and* by following the relationship “Unit of”

Thus, the supporting relationship needs to be many-to-one
Weak Entity Sets

• In class exercise:
  • Develop a university grading roster DB as an E/R diagram
  • You have courses and students as entities and enrollment as a connecting entity
  • Enrollment can have grade as an attribute
Weak Entity Sets

Students

- id number
- name
- address

Enrollment

- grade

course

- course number
- Department
- Term
- Timeslot
- Instructor

enrolled in

enrolled in
• Every enrollment record needs to have exactly a student and exactly a course
Weak Entity Sets

• In class exercise
  • Draw E/R diagrams involving weak entity sets
    • Courses and Departments
      • A course is given by a unique department, but its only attribute is its number
      • Different departments can offer courses with the same number
Weak Entity Sets

- Department
  - name
  - address
- course
  - course number
- teaches

The diagram shows the relationship between the Department entity and the course entity, where Department has attributes name and address, and course has the attribute course number.
From E/R Diagrams to Relational Design

- Each entity set becomes a relation with the same set of attributes
- Each relationship becomes a relation with attributes being the keys for the connected entity set
From E/R Diagrams to Relational Design

• Problems:
  • Weak entity sets cannot be translated straightforwardly
  • Isa relationships are difficult
  • Sometimes, makes sense to combine relations when connected by a many-to-one relationship
From E/R Diagrams to Relational Design

• In class test:

```
name  address
Stars

name  address
Studios

Stars-in

Studios

Movies

own

Stars

Movies

name
title
year
length
genre
```
From E/R Diagrams to Relational Design

stars(name, address)  movies(title, year, length, genre)

studios(name, address)  starsIn(name, title, year)

owns(name, title, year)
From E/R Diagrams to Relational Design

• In class exercise
From E/R Diagrams to Relational Design

stars(name, address)    movies(title, year, length, genre)

studios(name, address)

contracts(name, title, year, studioOfStar, producingStudio)
From E/R Diagrams to Relational Design

• Handling weak entity sets
  • Relation for a weak entity set needs to include key attributes of supporting entity sets
  • Relation for any relationship that includes a weak entity set must use as a key all of its key attributes, including those in supporting entities
  • A supporting relationship does not need to be represented in an entity itself
    • This is because the rule for the weak entity already force it to have these relationships
From E/R Diagrams to Relational Design

• Example
From E/R Diagrams to Relational Design

• Example

- **studios**(*name*, *address*)
- **crews**(*number*, *chief*, _studioName_)
- **unitOf**(*number*, _studioName_, *name*)

![E/R Diagram](image_url)
From E/R Diagrams to Relational Design

• Example

- studios \( \text{name, address} \)
- crews \( \text{number, chief, studioName} \)
- unitOf \( \text{number, studioName} \)

• Second pick: studioName and name are the same
From E/R Diagrams to Relational Design

- Example

  - Studios
    - name, address

  - Crews
    - number, chief

- Final pick: can dispense with unitOf

  - studios(name, address)
  - crews(number, chief, studioName)
From E/R Diagrams to Relational Design

• Converting subclass structures to relations
  • is-a relationship:
    • There is a root entity
    • Root entity has a key that identifies all entities in the hierarchy
    • A given entity may have *components* that belong to the entity sets of any subtree of the hierarchy that includes root
From E/R Diagrams to Relational Design

• Converting subclass structures to relations
  • Three strategies
    • Follow the E/R viewpoint
    • Treat entities as objects belonging to the same class
    • Use null values
From E/R Diagrams to Relational Design

- Follow the E/R view
- Make a relation for each entity
From E/R Diagrams to Relational Design

- movies(title, length, year, genre)
- murderMysteries(title, length, weapon)
- cartoons(title, year)
From E/R Diagrams to Relational Design

• E/R view:
  • movies(title, length, year, genre)
  • murderMysteries(title, length, weapon)
  • cartoons(title, year)

• A cartoon has a tuple in two tables
• “Who framed Roger Rabbit” has tuples in all three tables
• Add
  • voices(starName, title, year)

• Would still have to retain cartoons relationship since we might have silent cartoons
From E/R Diagrams to Relational Design

- Object-Oriented Approach to subclasses
  - Entities can only belong to one class
  - Enumerate all possible subtrees of the hierarchy
  - Create a relationship for all of them
From E/R Diagrams to Relational Design

- movies\((title, year, length, genre)\)
- moviesC\((title, year, length, genre)\)
- moviesMM\((title, year, length, genre, weapon)\)
- moviesCMM\((title, year, length, genre, weapon)\)
- A movie is in only one relationship
From E/R Diagrams to Relational Design

- movies(title, year, length, genre)
- moviesC(title, year, length, genre)
- moviesMM(title, year, length, genre, weapon)
- moviesCMM(title, year, length, genre, weapon)

Add voices:

- voices(title, year, starName)

Should we have two (depending on CMM or C)?

- Probably not, no good reason at this point
From E/R Diagrams to Relational Design

- Using Null values
  - Only have the root relation, but add to it all attributes in the hierarchy
    - movies(title, year, length, genre, weapon)
  - Use null value when movie not in MM
From E/R Diagrams to Relational Design

• Comparison of approaches
  • It can be expensive to answer queries involving several relations
    • “Null Value” approach wins
  • Different queries favor different set ups
    • What films of 2008 were longer than 150 minutes?
      • E/R approach is easy
      • OO approach needs to access four different relations
From E/R Diagrams to Relational Design

• Comparison of approaches
  • Different queries favor different set ups
    • “What weapons were used in cartoons over 120 minutes?”
      • OO approach needs to access one relation, moviesCMM
      • E/R approach: Access movies to find movies over 120 minutes, then access cartoons to see whether the movie is a cartoon, then access murderMysteries to find out the weapon
From E/R Diagrams to Relational Design

• We want few relations
  • “Null” approach works best
  • OO approach is worst
From E/R Diagrams to Relational Design

- We want to minimize space and avoid repetition
  - Null approach avoids repetition but tuples can now be very long
  - E/R approach: repeats data
  - OO approach: uses one tuple per entity
From E/R Diagrams to Relational Design

- Use E/R, OO, and Null approach on

```
Person
  ↓
Child
  ↓
ChildOf

Person
  ↓
Father
  ↓
FatherOf

Person
  ↓
Mother
  ↓
MotherOf

name

address

isa

isa

isa

isa

Married
```
Unified Modeling Language

- Developed as graphical notation for OO software design
## Unified Modeling Language

<table>
<thead>
<tr>
<th>UML</th>
<th>E/R</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>entity set</td>
</tr>
<tr>
<td>association</td>
<td>binary relationship</td>
</tr>
<tr>
<td>association class</td>
<td>attributes on a relationship</td>
</tr>
<tr>
<td>subclass</td>
<td>Is-a hierarchy</td>
</tr>
<tr>
<td>aggregation</td>
<td>many-one relationship</td>
</tr>
<tr>
<td>composition</td>
<td>many-one relationship with referential integrity</td>
</tr>
</tbody>
</table>
Unified Modeling Language

- UML classes
  - Classes:
    - 3 field box
      - name
      - instance variables (attributes)
      - bottom: methods
    - Used only in OO relational databases
Unified Modeling Language

- UML keys
  - Add PK (primary key) after attribute

<table>
<thead>
<tr>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>title PK</td>
</tr>
<tr>
<td>year PK</td>
</tr>
<tr>
<td>length</td>
</tr>
<tr>
<td>genre</td>
</tr>
<tr>
<td>&lt;place for methods&gt;</td>
</tr>
</tbody>
</table>
Unified Modeling Language

- Binary relationships are called *associations*
- No multiway relationships in UML

```
<table>
<thead>
<tr>
<th>studios</th>
<th>0..1</th>
</tr>
</thead>
<tbody>
<tr>
<td>name PK</td>
<td></td>
</tr>
<tr>
<td>address</td>
<td></td>
</tr>
<tr>
<td>owns</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>movies</th>
<th>0..*</th>
</tr>
</thead>
<tbody>
<tr>
<td>title PK</td>
<td></td>
</tr>
<tr>
<td>year PK</td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>0..*</td>
</tr>
<tr>
<td>genre</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stars-in</th>
<th>0..*</th>
</tr>
</thead>
<tbody>
<tr>
<td>stars</td>
<td></td>
</tr>
<tr>
<td>name PK</td>
<td></td>
</tr>
<tr>
<td>address</td>
<td></td>
</tr>
</tbody>
</table>
```
Unified Modeling Language

• Write down numerical restriction on associations at the other end
  • 0..1 at most one
    • a movie has at most one studio
  • 0..* any number
    • A studio owns any number of movies
    • A movie has any number of stars
    • A star has any number of movies
  • No label means: 1..1 (exactly one)
Each studio has to have at least one movie it owns.
Each movie is owned by exactly one studio.
Each president runs exactly one studio.
Each studio has one or none president.
Unified Modeling Language

- Each movie can have none, one, or more sequels
- Each movie can be the sequel of no movie (it’s not a sequel) or one movie (it is a sequel)
Unified Modeling Language

- Association classes

- There is no PK for compensation, the PK will be provided by the objects that are associated
Unified Modeling Language

- Subclasses in UML
  - UML allows four subclass relationships
    - Complete versus partial
      - Is every object a member of a subclass?
    - Disjoint versus overlapping
      - Can an object be in two sub-classes?
Unified Modeling Language

- Subclass objects **inherit** attributes from the superclass

- The relationship is disjoint, but only partial
Unified Modeling Language

- Aggregation (diamond) — many to one association
- Composition (filled diamond) — one-to-one association

Example:
- Every movie can be associated with at most one studio
- Every president has to have a studio, but not more than one
Unified Modeling Language

- Equivalent of weak entities:
  - Not necessary: In UML objects have their own attributes
  - Can use label PK in a composition
Unified Modeling Language

- Translating into Relations
  - Each class converts into a relation
  - Each association converts into a relation with the key attributes of the two connected classes
    - Renaming might be necessary
    - If there is an association class, relation also has attributes of the association class
Unified Modeling Language

- **studios** (name, address)
- **movies** (title, year, length, genre)
- **stars** (name, address)
- **owns** (studioName, movieTitle, movieYear)
- **stars-in** (starName, movieTitle, movieYear)
Unified Modeling Language
Unified Modeling Language

- Subclasses
  - Same possibilities as before:
    - Entity/Relationship approach
    - OO approach
    - Null values
Use E/R approach
movies(title, year, length, genre)
studios(name, address)
movieExecs(cert#, name, address, networth)
presidents(cert#)
presides(cert#, studioName)
owns(studioName, movieTitle, movieYear)
Unified Modeling Language

movies(title, year, length, genre)
studios(name, address)
movieExecs(cert#, name, address networth)
presidents(cert#)
presides(cert#, studioName)
owns(studioName, movieTitle, movieYear)

- Obviously, the presidents relation is superfluous
- What about owns?
Unified Modeling Language

- Dealing with compositions and associations:
  - They are many-to-one relationships
    - Incorporate the target relation into the other
    - If an aggregation, there might be no additional attributes
Unified Modeling Language

- This leads to a simpler database scheme

movies(title, year, length, genre, studioName)
studios(name, address)
movieExecs(cert#, name, address, netWorth)
presides(cert#, studioName)