**Database Management Systems**

**Course 236363**

**Lecture 2:**

**Entity-Relationship Diagrams**

---

**Outline**

- Introduction
- ER Diagrams
  - Entities
  - Relationships
  - Weak Entities
  - Type Hierarchies
- Translating ERD to Relational Schemas
- Design Principles

---

**Modeling Data**

- App development is often based on a formal modeling of the underlying data semantics
- Typically: **entities** of various **types**, connected by **relationships** of various **types**
- Examples:
  - Movies, actors, directors, roles, awards
  - Students, courses, lecturers, rooms
  - Products, users, purchases, credit companies
  - Dishes, ingredients, cooking actions
  - Divisions, battalions, soldiers, tanks, planes
  - Persons, statuses, friendships, messages, likes

---

**About Entity-Relationship Diagrams**

- Visual language (diagrams) for modeling data via entity types, relationship types, and constraints
  - But, does not necessarily cover all logic and constraints
- Have a formal and precise meaning
  - Need to thoroughly understand it to correctly design and interpret diagrams
- Middleman between logical layer and reality
  - Facilitates the process of defining the logical level of the data model (e.g., relational schemas)
  - Translates informal requirements into formal ones
- An opportunity to introduce elementary DB concepts
- Comes in many variants
  - Differ in visuals and semantics
  - We will use Garcia-Molina, Ullman & Widom

---

**Entity-Relationship Diagram**

- In short, **ER diagram** or **ERD**
- Graphical language / formalism to specify modeling of data in real-world scenarios
- What is it modeling?
  - **Entity** types
  - **Attribute** names per entity type
  - **Relationship** types
  - **Attribute** names per relationship type
  - **Constraints** on legal instantiations (actual sets of entities and relationships)
    - Movie has unique identifier and ≥1 directors; each award is associated with a unique movie, etc.

---

**Types, Instantiations, Sets**

- ERD does not talk about individual entities & relationships; only **types**
- An **instantiation** (or **instance**) of an ERD is a collection of **entity-sets** and **relationship-sets** of the corresponding types
  - For each type
- **Constraints** apply to the instantiations— which is legal and which is not?
Terminology so far...

- Entity
- Entity Type
- Entity Set
- Relationship
- Relationship Type
- Relationship Set
- Instances / instantiations
- Constraints

Outline

- Introduction
- ER Diagrams
  - Entities
  - Relationships
  - Weak Entities
  - Type Hierarchies
  - Translating ERD to Relational Schemas
- Design Principles

Graphical Components of ERDs

- Shapes (labeled w/ text) w/ different edge types
  - rectangle
  - ellipse
  - rhombus
  - triangle
- Connecting lines/arrows

ERD Example

Entities

- Each DB entity has an entity type with a name and a set of attribute names
- Denoted by a rectangle connected to ellipses (attributes)

An instance has a set of actors, each having an id, a birthday, a photo, ...
Attributes May Be Compound

One level; no more

 Attributes
 - day
 - month
 - year
 - birthday

Attributes
 - id
 - photo
 - name
 - address
 - first
 - middle
 - last

The Constraint of a Key Attribute

- Let:
  - $E$ be an entity type
  - $A = \{a_1, ..., a_k\}$ be a subset of the attributes of $E$
- By saying that $A$ is a key for $E$ we require that for every entity set $S$ of $E$, no two distinct entities have the same values for all attributes of $A$
- In notation:
  $$\forall e, f \in S \left[ (e[a_1] = f[a_1]) \land ... \land (e[a_k] = f[a_k]) \implies e = f \right]$$
- Hence, by specifying key attributes we specify both attributes and constraints (on the corresponding entity set)

Keys in ERD

- In ERD, underline names of attributes that constitute the key (if one exists)

Other Options Make Sense?

- Too restricted on address, too permissive on id
- Too restricted on name/emp# too permissive on id

What is the Difference?

Can we say that both id and emp# are keys?

No! (There is a limit to what can be expressed with a small set of shapes)

Multi-Value Attributes

Each actor may have multiple photos
Outline

- Introduction
- ER Diagrams
  - Entities
  - Relationships
  - Weak Entities
  - Type Hierarchies
  - Translating ERD to Relational Schemas
- Design Principles

Relationships

- By a relationship we mean a named association among entities
  - actsIn, directedBy, marriedTo, follows, messageAuthor, worksIn, ...
- A relationship type has a name and a set of entity types that participate in relationships
  - And possibly attribute names
- An instantiation consists of a set of relationships among entities of the corresponding types/sets
- As usual, a relationship constraint applies to the set of relationships of the corresponding type in an instance of the diagram

Relationship Examples

---

**Actors and movies relate to each other via the binary plays-in relationship**

- **Actor**
  - id
  - birthday
  - name

- **Movie**
  - id
  - name
  - genre
  - year

- **PlaysIn**
  - role

**An actor can play in any number of movies (including zero)**

**A movie can have any number of actors (including zero)**

---

Relationship Attributes

---

- **Actor**: id, birthday, name
- **Movie**: id, name, genre, year
- **PlaysIn**: role

Each plays-in relationship is associated with a role

Implicit constraint: No two relationships differ only in attributes!

(that is, the involved entities form a key for the relationship)

What should we do if we want an actor to have multiple roles?

---

Relationship Attributes

---

- **Actor**: id, birthday, name
- **PlayIn**: role
- **Movie**: id, name, genre, year

What should we do if we want an actor to have multiple roles?

---

More Than 2 Entity Types

---

- **Client**: name
- **Purchase**: id, name, vendor
- **Store**: address
- **Employee**: name, manager
- **Manager**: id, birthday

Ternary relationship type
Multiplicity Constraints on Rel. Sets

- Multiplicity constraints on relationship sets over types $E_1, ..., E_k, F$ involve the following:
  - Maximum # $F$ entities per combination of $E_1, ..., E_k$ entities
  - Minimum # $F$ entities per combination of $E_1, ..., E_k$ entities
- Graphically, denoted by decorating the edges between entity types and relationship types

Many-to-Many

A and B are in a many-to-many relationship if each B may have many A, but each A may have any number of Bs

Many-to-One

A and B are in a many-to-one relationship if each B may have many A, but each A may have at most one B

One-to-Many

A and B are in a one-to-many relationship if each B has at most one A, but each A may have many B

One-to-One

A and B are in a one-to-one relationship if each B has at most one A, AND each A has at most one B

Multiplicity in Multiway Relationships

What does it mean?

For every movie and role there is a single actor
(Put differently, Movie and Role determine Actor)
Limitation in Expressiveness

What does it mean?
• Movie and President (combined together) determine Studio
• Studio and Movie (combined together) determine President

In reality, Movie alone determines Studio; Studio alone determines President; ...
This is a limitation in ERD expressive power; typical in visual models, since there is only so much we can represent with arrows...

(Unique) Referential Integrity

• An R relates to precisely one L
• Here, every movie is owned by at most one studio, and moreover, every movie is owned by at least one studio
• (But a studio may exist without owning any movie, and a studio may own multiple movies)

Degree Constraints

Generalize X-to-X and referential constraints using explicit constraints written in math

A movie cannot have more than 2 stars

A movie has at least two actors

Roles in Relationships

• Sometimes an entity type participates more than once in a relationship
  • (e.g., ParentOf, Follows, HasALinkTo, ...)
  • To distinguish between the different roles of the entity type, we label each edge with a role name

Grouping a Relationship

By grouping a relationship type, we can treat whole relationships as entities (that participate in other relationships)
**Outline**

- Introduction
- ER Diagrams
  - Entities
  - Relationships
  - Weak Entities
  - Type Hierarchies
- Translating ERD to Relational Schemas
- Design Principles

**Weak Entity Types**

- Represent entities that are part of others
  - Departments of stores
  - Companies of battalions
  - Rooms of buildings
- We would like to say that a dept. is identified by its name (e.g., “pharmacy”) while allowing different stores to have depts. with that name
  - That is, we view different stores as independent for the matter of identifying departments

**More Formally**

- A weak entity type has an identifying owner connected by an identifying relationship
- Visually distinguish the weak entity and identifying relationship by double-edge shapes
- In a legal instance:
  1. Each weak entity is connected through the identifying relationship to precisely one owner
  2. The key of the weak entity is unique among those with the same owner; outside owner, key can repeat
     - Hence, the “real key” is the combination of the weak entity’s key and the owner’s key
**Identifying Keys**

- Crew
- WorksFor
- Studio
- name

What info uniquely identifies a crew? Its name and owner studio's name

Can there be two crews with the same name? Yes! In different studios...

---

**What's the Difference?**

- Company
- BC
- Battalion

- Company
- BC
- Battalion

- Company
- BC
- Battalion

**Outline**

- Introduction
- ER Diagrams
  - Entities
  - Relationships
  - Weak Entities
  - Type Hierarchies
- Translating ERD to Relational Schemas
- Design Principles

---

**ISA Relationships**

- ISA is a special relationship used for representing subtypes or subclasses
- Examples:
  - Director is a Movie Person, who is a Person
  - Cartoon is a Movie; Action-Movie is a Movie
  - Engineer is a Employee
- Important difference from OOP: an entity (object) can be of different subtype (subclasses) at the same time
  - For instance, a cartoon action movie
  - Closer to interfaces in Java

---

**Representing ISA**

What is the key of an Actor? A director?

Person's id! Subclass must respect class and cannot redefine keys!
On ISA

- Every entity of B is also of A
- Every entity of C is also of A
- There may be entities that are of both B and C types
- There may be entities of A that are of neither B nor C

(There are ERD formalisms that allow to distinguish between these cases, see next...)

Specialization and Generalization

Specialization

- All Bs are As
- Some As are Bs

Generalization

- Some As are C's
- Every A is either a B or a C

Outline

- Introduction
- ER Diagrams
  - Entities
  - Relationships
  - Weak Entities
  - Type Hierarchies
- Translating ERD to Relational Schemas
- Design Principles

The Relational Database Model

- A relational database is modeled via two concepts:
  - A relational schema
    - Spec of structure, types, constraints
  - A database instance over the schema
    - Actual tables (relations) with actual rows (tuples) and values (attribute values) corresponding to the schema

Relational Schema

- Just schema for short
- Consists of:
  - A signature
    - Relation names and associated attributes (names/types)
  - Constraints on the signature
    - Attribute constraints (uniqueness, legal range, ...)
    - Relation constraints (e.g., cardinality)

Signature Example

Movie(title, year, length, genre, rating)

- Relation name: Movie
- 5 attributes: title, year, length, gender, rating
- Attributes have domains (sets of legal values)
- We often ignore the domains if they are irrelevant to the discussion
  - Or if they significantly complicate things
Examples of Constraints: Key Constraint

- A set $K$ of attributes such that no two distinct tuples can have the same values on every attribute in $K$
- Example: "no two Movie tuples can have both the same title and the same year"
  - Hence, $(title, year)$ is a key for Movies
  - In the common case where there is a single key, we denote the key attributes using underline:
    
    \[
    \text{Movie}(title, year, length, genre, rating)
    \]

Examples of Constraints: Foreign Key

- A set $F$ of attributes is a foreign key of a relation $R$ if there is a relation $S$ with a key $K$ such that for every $r$ in $R$ there is $s$ in $S$ such that $r[F]=s[K]$
  - Note: $t(X)$ is obtained from $t$ by restriction to $X$
- In $\text{Role}(actor, movie, role)$, the $actor$ value must be the $id$ key of a tuple in $\text{Actor}(id, name, photo)$
  - In our notation, we will use arrows
    
    \[
    \text{Role}(actor, movie, role) \quad \text{Actor}(id, name, photo)
    \]
  - Applies to multi-attribute keys (e.g., firstName, lastName)
  - (Later in the course we will get deeper into schema constraints)

ERD to Relational Schema

- Context:
  - We have an ERD for our application data
  - We wish to store our data in a relational DB
  - Need to convert: ERD $\rightarrow$ relational schema
- Principles:
  - Avoid duplicating information
  - Constrain as much as possible
    - We should be able to map legal schema instances back to the ER model without violating any ERD constraint

Translating an Entity Type

- Straightforward
  - entity name $\rightarrow$ relation name
  - attribute name $\rightarrow$ attribute name
  - key $\rightarrow$ key

Set Attribute

- $\text{Actor}(id, name, birthday, address)$
- $\text{Photos}(aid, photo)$

Example of Relationship Translation

- $\text{Role}(actor, movie, role)$
- $\text{Actor}(id, name, birthday)$
- $\text{Movie}(name, year, genre)$
Translating a Relationship

- Translation:
  - relationship name $\rightarrow$ relation name
  - entity keys + relationship attributes $\rightarrow$ relation attributes
- Attributes may need to be renamed for distinctness and clarity
- Entity keys form the key of the new relation

Example in PostgreSQL

```
CREATE TABLE Actor (
    id int,
    name text,
    birthday date,
    PRIMARY KEY (id)
)
CREATE TABLE Movie (
    name text,
    year int,
    genre text,
    PRIMARY KEY (name,year)
)
CREATE TABLE PlaysIn (
    aid int,
    name text,
    year int,
    salary int,
    PRIMARY KEY (aid,name,year),
    FOREIGN KEY (aid) REFERENCES Actor(id),
    FOREIGN KEY (name,year) REFERENCES Movie(name,year)
)
```

Translating a One-to-Many Relationship

```
Studio(name,address) Owns Movie(name,year,genre)
```

Translate:

- Remove attributes from the relationship key

```
Owns(sname,mname,year)
```

```
Studio(name,address) Movie(name,year,genre)
```

Translating Unique Reference

```
Studio(name,address) Owns Movie(name,year,genre)
```

Translate:

- Add a foreign key

```
Movie(name,year,genre,sname)
```

```
Studio(name,address)
```

Could we do it with the relation Owns?

```
Owns(sname,mname,year)
```

```
Studio(name,address) Movie(name,year,genre)
```

Translating a One-to-Many Relationship

```
Studio(name,address) Owns Movie(name,year,genre)
```

Translate:

- Remove attributes from the relationship key

```
Owns(sname,mname,year)
```

```
Studio(name,address) Movie(name,year,genre)
```
Translating Weak Entities

- Recall: a weak entity has an identifying relationship to an identifying entity
- Typical translation:
  - No specific relationship for the identifying relationship
  - Weak entity references its identifying entity
  - Key of weak entity includes the key of its identifying entity

Example

![Diagram showing Crew, WorksFor, and Studio entities]

Translating ISA

- Similar to weak entities: subclass references superclass

Outline

- Introduction
- ER Diagrams
  - Entities
  - Relationships
  - Weak Entities
  - Type Hierarchies
- Translating ERD to Relational Schemas
- Design Principles

Recipe for ERD Design

1. Identify the proper entity types
2. Determine if there are hierarchies (ISA or weak relationships) among entity sets
3. Identify the proper relationship types
4. Identify the attributes and keys
5. Determine relationship constraints

Good Practices

- Faithfulness
- Non-Redundancy
- Simplicity
Faithfulness (1)
The design should correctly model the requirements of the application

Actor PlaysIn salary Movie

What could be a problem?

Actor’s salary changes from one movie to another; it is not an attribute of the actor, but rather of the relationship!

Faithfulness (2)
Make sure that relationship types make as accurate associations as possible – constraints used precisely when needed

Agency Reg Actor ActsIn Actor ActsIn Agency
Agency Reg Actor ActsIn Movie

Which is correct? Depends on the application!

Question

• Scenario 1:
  – An Employee belongs to a single project
  – For Project work, employees use Tools

• Scenario 2:
  – An employee belongs to a single project
  – An employee may use each tool for just one project

• Scenario 3:
  – An employee belongs to a single project
  – An employee may use a tool only for its owner project

Answer in class discussion…

Non-Redundancy
Avoid representing information that can be inferred otherwise (resulting in larger and slower databases, complicates maintenance, raising the risk of inconsistency)

name Studio Owns Movie studioName

What’s the problem?

studioName is redundant

Non-Redundancy
Avoid representing information that can be inferred otherwise (resulting in larger and slower databases, complicates maintenance, raising the risk of inconsistency)

name Studio Owns Movie studioName

What’s the problem?

all attributes follow from studioID; need to repeat for each movie
Simplicity (1)

Simpler is better!
Avoid introducing unneeded modeling and complexity

Do we need the Property entity?
Depends on the app: movies, real-estate, ...

Simplicity (2)

Simpler is better!
Avoid introducing unneeded modeling and complexity

What about Mammals?
Does it help to distinguish between movie persons and non-movie persons?

What about Drama Actors?
All depend on the application needs. Do we have mammals beyond people? Do drama actors have special attributes? …