Database Management Systems
Course 236363

Lecture 2:
Entity-Relationship Diagrams

Outline

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

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

The IMDb Application

Steps in Database Setup

- Requirement analysis
  - What information needs to be stored?
  - How will it be used?
  - What integrity constraints should be imposed?
- Conceptual database design
  - Define/describe/discuss the semantic modeling of data in the application (ER model via ER diagrams)
- Logical database design
  - Translate the ER diagram into a relational DB schema
- Physical database design
  - Translate the database schema into a physical storage plan on available hardware (done by DBMS)

Entity-Relationship Diagram (ER Diagram / ERD)

- Formalism to model 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 (sets of entities and relationships)
- A movie has a unique identifier and ≥ 1 directors; each award is associated with a unique movie, etc.
- No individual entities & rels.; only types!
  - Instantiations consist of sets of entities and sets of relationships of the corresponding types
Terminology so far...

- Entity
- Entity Type
- Entity Set
- Relationship
- Relationship Type
- Relationship Set
- Instances

More on ERD

- Presented and taught by a visual language (diagrams) rather than a textual one
- 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

Outline

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

Graphical Components of ER Diagrams

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

ERD Example

Outline

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

• An entity type has 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

Attributes May Be Compound

Key Attributes

• Let $E$ be an entity type and $A$=$\{a_1, \ldots, a_k\}$ a subset of the attributes of $E$
• We say that $A$ is a key for $E$ if for every legal set $S$ of $E$ entities, no two distinct entities have the same values for $A$
• In notation:
  $$\forall e, f \in S (e[a_1]=f[a_1] \land \ldots \land e[a_k]=f[a_k]) \Rightarrow e=f$$
• Hence, by specifying key attributes we specify both attributes and constraints
  – Constraints are on entity sets/instances

Keys in ERD

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

It is conventional to have a key for every entity type (unless we have a good reason not to); keys are sometimes “natural” (e.g., SSN) and sometimes artificial (internal identifiers)

Other Options Make Sense?

Other Options Make Sense?

What is the Difference?

What is the Difference?

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

There is a limit to what we can express with a small set of arrows and shapes.
Multi-Value Attributes

Each actor may have multiple photos

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

• 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
  • An actor can play in any number of movies (including zero)
  • A movie can have any number of actors

Relationship Attributes

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)

More Than Two Entity Types

What should we do if we want an actor to have multiple roles?
Multiplicity Constraints on Rel. Sets

- Multiplicity constraints on relationship sets over entities $E_1, \ldots, E_k$ involve the following:
  - Maximum # $F$s per $E_1, \ldots, E_k$
  - Minimum # $F$s per $E_1, \ldots, E_k$

- Graphically, denoted by decorating the edges between entity types and relationship types

Many-to-Many

- An $L$ can relate to any number of $R$s
- An $R$ can relate to any number of $L$s

Many-to-One

- An $L$ can relate to at most one $R$
- An $R$ can relate to any number of $L$s

One-to-Many

- An $L$ can relate to any number of $R$s
- An $R$ can relate to at most one $L$

One-to-One

- An $L$ can relate to at most one $R$
- An $R$ can relate to at most one $L$

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)

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

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

![Diagram of Studio Owns Movie](image)

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

![Diagram of Studio Owns Movie](image)

- 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)

Which Graphs Match This Meaning?

![Graphs A to F](image)

Answer: C, E

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
  - Design Principles
- Translating ERD to Relational Schemas

---

**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., “kids”) while allowing different stores to have departments with that name
  - That is, we view different stores as independent for the matter of identifying departments
- In ERD, we distinguish the subentity-entity relationship by double-edge shapes

---

**Identifying Keys**

- Can there be two crews with the same name (e.g., Crew 1)?
- Makes sense within a studio; but outside?

---

**Examples of Weak Entities**

- Company
  - BC
  - Battalion
- “Weak Entity” number
- “Identifying Relationship” number
- “Identifying owner” name

---

**Identifying Keys**

- Can there be two crews with the same name?
- What info uniquely identifies a crew?
- Answer: name + studio name
- Answer: yes

---

**What’s the Difference?**

- Company
  - BC
  - Battalion
  - “Weak Entity” number
  - “Identifying Relationship” number
  - “Identifying owner” name
What's the Difference? Answer

- Each company has a unique battalion
- In each battalion, company numbers are distinct

- Each company has a unique battalion
- Company numbers are distinct across DB

- A company may have any number of battalions
- Company numbers are distinct across DB

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 an Employee
- Important different from OOP: an object can be of different subclasses at the same time
  - For instance, a cartoon action movie

Representing ISA

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...)

Outline

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

Faithfulness (2)

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

Non-Redundancy

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

Simplicity (1)

Simpler is better!
Avoid introducing unneeded modeling and complexity

Do we need Property entity? Depends...
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?

Salary

ID

MoviePerson

Actor

Director

Introduction

ER Diagrams

- Entities
- Relationships
- Weak Entities
- Type Hierarchies
- Design Principles

Translating ERD to Relational Schemas

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
    - Which combinations of relations are allowed in schema instances?

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:
    `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)
  
• In Role(actor, movie, role), the actor value must be the id key of a tuple in Actor(id, name, photo)
  
• (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 → relational schema

• Principles:
  
  – Avoid duplicating information
  
  – Constrain as much as possible
    
    • Ideally, 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 → relation name
  
  – attribute name → attribute name
  
  – key → key

  ![Diagram: Translating an Entity Type]

  Actor(id, name, birthday, photo, address)

Set Attribute

![Diagram: Set Attribute]

Actor(id, name, birthday, photo)

Photos(id, photo)

Example of Relationship Translation

![Diagram: Example of Relationship Translation]

PlaysIn(aid, name, year, salary)

Actor(id, name, birthday)  Movie(name, year, genre)

Translating a Relationship

• Translation:
  
  – relationship name → relation name

  – entity keys + relationship attributes → 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)
```

Remove attributes from the relationship key

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

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

Translating Unique Reference

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

Add a foreign key

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

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

```
Could we do it with the Owns Relationship?
```

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

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

Crew(name, sname, room)
Studio(name, address)

Translating ISA

• Similar to weak entities: subclass references superclass

Person(id, name, birthday)
Actor(id, photo)
Director(id)

ISA