Database Technology
Entity Relationship Models
Previously on Database Technology

• Introduction to Relational Databases
  – A standard model for storing data
  – Using relations/tables

• Introduction to SQL
  – Creating and changing tables
  – Reading and writing data into tables
Today

- Designing databases
  - i.e., how to get from your customer’s requirements…
  - ...to a set of tables and attributes
Outline

• Design Process
• Modeling
• Constraints
• E-R Diagrams
• Design Issues
• Weak Entity Sets
• Extended E-R Features
• Reduction to Relation Schemas
• Comparison UML
Database Design

• Initial phase: requirements engineering
  – characterize fully the data needs of the prospective database users
  – which data needs to be stored?
    • ...and in which volumes?
  – which queries should be answered?

• Conceptual schema
  – which types of entities and relations exist?
  – what attributes do they have?
Database Design

• Final phase: from a conceptual to physical data model
  – Logical Design: find a “good” collection of relation schemas
    • Business decision – What attributes should we record in the database?
    • Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
  – Physical Design – Deciding on the physical layout of the database
Database Design Approaches

• Entity Relationship Model (today)
  – Models an enterprise as a collection of entities and relationships
  – Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
    • Described by a set of attributes
  – Relationship: an association among several entities
  – Represented diagrammatically by an entity-relationship diagram

• Normalization Theory (next lecture)
  – Formalize what designs are bad, and test for them
Entity Relationship Model

• Dates back to the 1970s
  – developed to facilitate database design by allowing the specification of an enterprise schema that represents the overall logical structure of a database

• Toolkit for mapping the meanings and interactions of real-world enterprises onto a conceptual schema

• The ER data model employs three basic concepts:
  – entity sets,
  – relationship sets,
  – attributes

• Associated diagrammatic representation (ER diagram)
  – graphic expression of the overall logical structure of a database
Entity Sets

• An entity is an object that exists and is distinguishable from other objects
  – Example: Peter Chen, Mannheim, Star Wars
• An entity set is a set of entities of the same type that share the same properties
  – Example: set of all persons, cities, movies
• Each entity is represented by a set of attributes
  – Example:
    - instructor = (ID, name, street, city, salary)
    - course= (course_id, title, credits)
• A subset of the attributes form a primary key of the entity set i.e., uniquely identifying each member of the set
Entity Sets – Example

- instructor (instructor_id, instructor_name)
- student (student_id, student_name)
Relationship Sets

• A **relationship** is an association among several entities

  Example:
  44553 (Peltier)  *advisor*  22222 (Einstein)
  *student* entity  relationship set  *instructor* entity

• A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

  $$\{(e_1, e_2, \ldots, e_n) | e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$
  where $(e_1, e_2, \ldots, e_n)$ is a relationship

• Example:

  $$(44553, 22222) \in advisor$$
Relationship Sets

• An attribute can also be associated with a relationship set
• E.g., advisor relationship set:
  – date which captures the start of the supervision
Degree of a Relationship

• Definition: degree of a relationship
  i.e., number of entity sets that are involved in relation set

• binary relationship (degree two)
  – involve two entity sets
  – the by far most frequent case

• Relationships between more than two entity sets (degree >2)
  – e.g.: students work on projects under the guidance of an instructor
  – relationship proj_guide is a ternary relationship
    between instructor, student, and project
  – those are rather rare
Cardinality Constraints

• Express the number of entities to which another entity can be associated via a relationship set
  – Most useful in describing binary relationship sets

• For a binary relationship set, the mapping cardinality must be one of the following types:
  – 1:1 (one to one)
  – 1:n (one to many)
  – n:1 (many to one)
  – n:m (many to many)
Mapping Cardinalities – One to One

- One to one (1:1)
  - Note: Some elements in A and B may not be mapped to any elements in the other set

- Examples
  - student_works_on_thesis
  - department_has_dean
Mapping Cardinalities – One to Many

• One to many (1:n)
  – Note: Some elements in A and B may not be mapped to any elements in the other set

• Examples
  – building_has_room
  – course_taught_by_lecturer
Mapping Cardinalities – Many to One

• Many to one (n:1)
  – Note: Some elements in A and B may not be mapped to any elements in the other set

• Examples
  – room_located_in_building
  – lecturer_teaches_course

• Note:
  – the inverse of a 1:n relation is a n:1 relation
  – and vice versa
Mapping Cardinalities – Many to Many

• Many to many (n:m)
  – Note: Some elements in A and B may not be mapped to any elements in the other set

• Examples
  – student_takes_course
  – student_has_advisor
Distinguishing 1:n/n:1 and n:m Cardinalities

• Rule of thumb
  – Always ask for the cardinality the other way around

• “A building may have multiple rooms...”
  – “…but can a room be in multiple buildings?”
  – No → building_has_room is 1:n

• “A department can be located in multiple buildings...”
  – “…but can a building host multiple departments?”
  – Yes → department_located_in_building is n:m
Relation Sets from the Same Entity Set

- The two entity sets in a relation set may be the same
- This holds independently from the cardinality!

- `person_married_to_person`
  - 1:1
- `person_is_father_of_person`
  - 1:n
- `person_has_father`
  - n:1
- `person_is_parent_of_person`
  - n:m
Attribute Types & Domains

• Attribute types:
  – Simple and composite attributes
  – Single-valued and multi-valued attributes
    • Example: multi-valued attribute: phone_numbers

• Derived attributes
  – Can be computed from other attributes
  – Example: age (given date_of_birth)

• Domain – the set of permitted values for each attribute
Composite Attributes

- **Composite attributes**
  - `first_name`
  - `middle_initial`
  - `last_name`

- **Component attributes**
  - `street_number`
  - `street_name`
  - `apartment_number`
  - `street`
  - `city`
  - `state`
  - `postal_code`
Redundant Attributes

• Suppose we have entity sets:
  – instructor, with attributes: ID, name, dept_name, salary
  – department, with attributes: dept_name, building, budget

• In ERM, instructors and departments are connected by a relation set
  – e.g., instructor_belongs_to_department (ID,dept_name)

• Now, dept_name is no longer needed in the instructor entity set
  – It is redundant there
  – Hence, we will remove it

• Note: sometimes, removed redundant attributes are reintroduced when converting the conceptual model into a logical model
Weak Entity Sets

- Consider the set of buildings and rooms
  - Entity set building(building_name,address)
  - Entity set room(number,capacity)
  - Relation set room_in_building (number,building_name)

- Note:
  - As in the previous example, we have removed the redundant attribute building_name from the entity set room

- Question:
  - What is the primary key of the the entity set room?
Weak Entity Sets

• Weak entity sets are entity sets that
  – do not have a set of attributes sufficient to identify each entity uniquely
  – require an additional relation set to identify each entity uniquely

• Those relation sets are called *identifying relation set*

• Weak entities do not have primary keys
  – A weak entity set has an *identifying entity* and a *discriminator*
  – Example:
    • *building* is the identifying entity
    • *number* is the discriminator

• A weak entity cannot exist without the identifying entity
  – e.g., a room cannot exist without the building
ER Diagrams

- Entity Relationship Diagrams (ER diagrams)
  - are the graphical notation of entity relationship models
- Notation of entity sets:
  - Rectangles represent entity sets
  - Attributes listed inside entity rectangle
  - Underlining indicates primary key attributes
ER Diagrams

- Diamonds represent relationship sets
ER Diagrams

- Diamonds represent relationship sets
  - Attributes can be attached to relationship sets

```
<table>
<thead>
<tr>
<th>instructor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>tot_cred</td>
</tr>
</tbody>
</table>
```
Roles

- Entity sets of a relationship need not be distinct
  - i.e., there may be a relationship set involving the same entity set twice
- Each occurrence of an entity set plays a “role” in the relationship
  - The labels “course_id” and “prereq_id” are called roles
Cardinalities in ER Diagrams

• We express cardinality constraints by drawing either a directed line (→), signifying “one”, or an undirected line (—), signifying “many”, between the relationship set and the entity set.

• One-to-one relationship between an instructor and a student:
  – A student is associated with at most one instructor via the relationship advisor
  – An instructor is associated with at most one student via the relationship advisor
Cardinalities in ER Diagrams

• one-to-many relationship between an *instructor* and a *student*
  – an instructor is associated with several (including 0) students via *advisor*
  – a student is associated with at most one instructor via advisor
Cardinalities in ER Diagrams

- Many to many relationships
  - An instructor is associated with several (possibly 0) students via advisor
  - A student is associated with several (possibly 0) instructors via advisor

```
<table>
<thead>
<tr>
<th>instructor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>tot_crd</td>
</tr>
</tbody>
</table>
```
Total and Partial Participation

- Total participation (double line)
  - every entity in the entity set participates in at least one relationship in the relationship set
  - i.e., every student must have an advisor
    - recap: think of not null constraints

- Partial participation (single line)
  - some entities may not participate in the relationship
  - e.g., not every instructor has to supervise a student
Complex Cardinality Constraints

• Notation for minimum/maximum cardinality of a relation
  – Each student has \textit{exactly one} advisor (i.e., min=max=1)
  – Each instructor can be the advisor of multiple students, but needs not be (i.e., min=0,max=\infty)

• Notation:
  – min..max
  – * indicates no limit
Notation of Attribute Types

instructor

- **ID**
- **name**
  - **first_name**
  - **middle_initial**
  - **last_name**
- **address**
  - **street**
    - **street_number**
    - **street_name**
    - **apt_number**
- **city**
- **state**
- **zip**
  - **phone_number**
- **date_of_birth**
- **age()**

- complex attribute
- multivalued attribute
- derived attribute
Expressing Weak Entity Sets

• A weak entity set is depicted via a double rectangle
  – The identifying relationship set is depicted by a double diamond
• The discriminator is underlined with a dashed line
  – Primary key for section – (course_id, sec_id, semester, year)
Higher Arity Relationship Sets

- Most relationship sets are binary
- Sometimes, ternary (or higher arity) relations occur
  - ER models support that
- Example:
  - Students work on projects under supervision of an instructor

![ER Diagram]

- Instructor
  - ID
  - name
  - salary
- Project
  - ... (Attributes not shown)
- Student
  - ID
  - name
  - tot_cred
- proj_guide (Relationship)

ER Diagram represents the relationships between instructor, student, and project.
Cardinality Constraints for Ternary Relations

- Only one single arrow (i.e., cardinality restriction) is allowed for a ternary relation
  - Example: each student can work in at most one project under the supervision of some instructor(s)
Cardinality Constraints for Ternary Relations

- Multiple single arrows (i.e., cardinality restrictions) would lead to different possible interpretations
  - Each student works on at most one project under at most one instructor
  - For each project a student works on, there is at most one instructor
  - For each instructor supervising a student, there is at most one project
- Hence, we do not allow for them
Specialization

- A concept very common in (object oriented) programming
  - Entity sets are sub-/super sets of others
  - They inherit all the attributes from their super sets

- Overlapping
  - A person can be both an employee and a student

- Disjoint
  - An employee can be either an instructor or a secretary
Partial vs. Total Specialization

- **Partial specialization**
  - An employee may be an instructor or a secretary, or an employee not further specified
  - the default case

- **Total specialization**
  - There are no other persons than employees and students (in the DB)
  - Needs to be specified in the diagram
  - Analogy in OOP: abstract classes
A Full Example
Reduction to Relation Schemas

• How to get to from an ER model to a relational database model?
  – Recap: relational database models consists of relations

• We have
  – Entity sets and relationship sets

• Goal
  – Translate entity and relationship sets uniformly to relation schemas

• Mechanism:
  – For each entity set and relationship set there is a unique relation that is assigned the name of the corresponding entity set or relationship set
  – Each relation has a number of columns (generally corresponding to attributes), which have unique names
Representing Entity Sets

- A strong entity set becomes a relation with the same attributes
  building(name, address)
- A weak entity set becomes a relation that includes
  - the column(s) of the primary key of the identifying strong entity set:
    room (name, number, capacity)
- At the same time, name is a foreign key
  - which integrity constraints should we use?
Representing Relationship Sets

- Many-to-many relationship sets
  - represented as a relation with attributes for the primary keys of the two participating entity sets
- Example: schema for relationship set *advisor*

\[
\text{advisor} = (\text{student}\_\text{ID}, \text{instructor}\_\text{ID})
\]
Representing Relationship Sets

• Many-to-many relationship sets
  – additional attributes of the relationship set become attributes of the representing relation

• Example: schema for relationship set *advisor*
  
  $$advisor = (student\_ID, instructor\_ID, date)$$
Representing Relationship Sets

• Special case for one-to-many relationship sets
  – The primary key of the “many” side can become a foreign key attribute on the “one” side
    
    \[
    \text{student} = (ID, \text{name, tot_cred, instructor\_ID})
    \]

• In case of partial participation, this may cause \text{null} values
Representing Relationship Sets

• Special case for one-to-one relationship sets
  - The primary key on one side can be included on the other side
    
    \[
    \text{student} = (ID, \text{name}, \text{tot\_cred}, \text{instructor\_ID}) \quad \text{or} \quad \\
    \text{instructor} = (ID, \text{name}, \text{salary}, \text{student\_id})
    \]

• In case of partial participation, this may cause \text{null} values
Representing Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute.
- Add prefix of super attribute in case ambiguous names occur. Example: `street_number`, `phone_number`.
- Ignoring multivalued attributes, extended instructor schema is:
  
  \[
  \text{instructor}(\text{ID, first}_\text{name}, \text{middle}_\text{initial, last}_\text{name, street}_\text{number, street}_\text{name, apt}_\text{number, city, state, zip}_\text{code, date}_\text{of}_\text{birth})
  \]
Representing Multi-valued Attributes

• A multivalued attribute $M$ of an entity $E$ is represented by a separate schema $EM$

• Schema $EM$ has attributes corresponding to the primary key of $E$ and an attribute corresponding to multivalued attribute $M$
  – Example: Multivalued attribute phone_number of instructor is represented by a schema:
    $inst\_phone=$ (ID, phone_number)

• Each value of the multivalued attribute maps to a separate tuple of the relation on schema $EM$
  – Example: an instructor entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:
    (22222, 456-7890) and (22222, 123-4567)
Representing Derived Attributes

- Technically, we can create a view

  ```sql
  create view instructor_age as
  select ID, NOW() - date_of_birth as age
  from instructor
  ```
Representing Higher Arity Relations

• Higher arity relationship sets are represented just like binary ones
  – i.e., as one relation with the primary keys of the related entity sets
  – `proj_guide(instructor_ID, student_ID, project_ID)`
Representing Specialization

• Method 1
  – All three relations become relations
    • primary key is shared
  – Shared attributes are only represented in the higher level entity
    person(ID, name, street, city)
    employee(ID, salary)
    student(ID, tot_credits)

• Drawback:
  – Accessing person information for employees and students requires access to two relations
Representing Specialization

• Method 2
  – All three relations become relations
    • primary key is shared
  – Shared attributes are only represented in each entity
    person(ID, name, street, city)
    employee(ID, name, street, city, salary)
    student(ID, name, street, city, tot_credits)
  – Super relation can be omitted for total specialization

• Drawback:
  – Redundant storage for overlapping specialization
    • i.e., for persons that are both employees and students
Design Decisions in ER Modeling

• Entity sets vs. attributes

  - Entity set
    - Allows for additional information
    - Supports multi-valued attributes
      - in that case, the attribute would end as a relation in the DB anyways
Entity Sets vs. Relationship Sets

- Students register for course sections
  - This could be a simple relationship set as well
- Entity set can store additional information, e.g.
  - Date of registration
Placement of Attributes for 1:1 Relationships

• The primary key on one side can be included on the other side
  
  \[
  \text{student} = (ID, \text{name, tot_cred, instructor\_ID}) \text{ or }
  \text{instructor} = (ID, \text{name, salary, student\_id})
  \]

• Which one?
Binary vs. Non-Binary Relationships

- Sometimes, non-binary relationships can be replaced by binary ones
Binary vs. Non-Binary Relationships

- Sometimes, non-binary relationships can be replaced by binary ones
  - This is usually the preferred solution

```
- has_mother
  - child

- person
  - name
  - birthday

- has_father
  - child
```
Binary vs. Non-Binary Relationships

- Sometimes, non-binary relationships can be replaced by binary ones
  - but sometimes, they are n-ary by nature
Binary vs. Non-Binary Relationships

- Sometimes, non-binary relationships can be replaced by binary ones
  - but sometimes, they are n-ary by nature

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>instr_</td>
<td>project</td>
<td>student_</td>
</tr>
<tr>
<td>instructor</td>
<td>ID</td>
<td>student</td>
</tr>
<tr>
<td>ID</td>
<td>name</td>
<td>ID</td>
</tr>
<tr>
<td>student</td>
<td>ID</td>
<td>tot_credits</td>
</tr>
<tr>
<td>ID</td>
<td>name</td>
<td></td>
</tr>
</tbody>
</table>
```
Binary vs. Non-Binary Relationships

- Sometimes, non-binary relationships can be replaced by binary ones
  - but sometimes, they are n-ary by nature
- General decomposition schema:
ER Design Decisions (Summary)

- The use of an attribute or entity set to represent an object
- Whether a real-world concept is best expressed by an entity set or a relationship set
- The use of a ternary relationship versus a number of binary relationships
- The use of a strong or weak entity set
- The use of specialization/generalization – contributes to modularity in the design
Summary of ER Notation

- **E**: entity set
- **R**: relationship set
- **E** with **R**: total participation of entity set in relationship
- **E** with **R** for weak entity set: identifying relationship set

**Attributes**:
- Simple (A1)
- Composite (A2)
- Multivalued (A3)
- Derived (A4)

**Primary Key**: A1
entity set E with simple attribute A1, composite attribute A2, multivalued attribute A3, derived attribute A4, and primary key A1
Alternative ER Notations (ctd.)

- **many-to-many relationship**
  - Diagram: E1 * R * E2
  - Diagram: E1 \rightarrow R \rightarrow E2

- **one-to-one relationship**
  - Diagram: E1 1 R 1 E2
  - Diagram: E1 \rightarrow R \rightarrow E2

- **many-to-one relationship**
  - Diagram: E1 * R 1 E2
  - Diagram: E1 \rightarrow R \rightarrow E2

- **participation in R: total (E1) and partial (E2)**
  - Diagram: E1 \rightarrow R \rightarrow E2
  - Diagram: E1 \rightarrow R \rightarrow E2
Alternative Modeling Paradigms: UML

- Unified Modeling Language
  - often used in software design
  - similar scope: objects and their relations
  - ISO standard since 2005

- ER models in RDBMS
  - Direct translation to SQL

- UML models in software engineering
  - Direct translation to source code
Alternative Modeling Paradigms: UML

**ER Diagram Notation**

<table>
<thead>
<tr>
<th>E</th>
<th>entity with attributes (simple, composite, multivalued, derived)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td></td>
</tr>
<tr>
<td>M10</td>
<td></td>
</tr>
</tbody>
</table>

**Equivalent in UML**

<table>
<thead>
<tr>
<th>E</th>
<th>class with simple attributes and methods (attribute prefixes: + = public, − = private, # = protected)</th>
</tr>
</thead>
<tbody>
<tr>
<td>−A1</td>
<td></td>
</tr>
<tr>
<td>+M10</td>
<td></td>
</tr>
</tbody>
</table>

- Binary relationship:
  - E1
  - R
  - E2
  - role1
  - role2

- Relationship attributes:
  - E1
  - R
  - E2
  - role1
  - role2

- Cardinality constraints:
  - E1
  - R
  - E2
  - 0..1
  - 0..1

---

3/13/23
Heiko Paulheim

70
Alternative Modeling Paradigms: UML

http://pld.cs.luc.edu/database/ER.html
Summary

• Designing databases
  – i.e., how to get from your customer’s requirements…
  – ...to a set of tables and attributes

• ER Models are an intermediate step
  – Conceptual view on the database
  – Graphical notation
  – Can be used for discussion with customers

• Translation rules for ER to RDBMS

• Design decisions
  – For ER Models (mostly business decisions)
  – For translation to RDBMS (mostly computer science decisions)
Questions?