Chapter 7: DB Design and the Entity-Relationship (ER) Model
DB Design Layers

- Like all complex designs, the design of a (large) DB must use abstraction, a.k.a. hierarchical approach, a.k.a. layers or phases

- Two main strategies:
  - Bottom-up
  - Top-down

Ch. 7 is devoted to bottom-up (while Ch. 8 is top-down)
3 levels of abstraction

DB modeling generally has 3 levels of abstraction, a.k.a. the “three-schema” architecture:

- Level 0: The real-world enterprise.
- Level 1: Conceptual Model, e.g. E-R model.
- Level 2: Logical or Implementation Model, e.g. relational model.
- Level 3: Physical Model, consisting of low-level Data Structures, e.g. red-black trees.
Phase 0

- Characterize as completely as possible the data needs of the customer (a.k.a. enterprise)

- DB designer must interact with (talk to) all future users of the DB (a.k.a. domain experts)

- Outcome: specification of user requirements
Phase 1: Conceptual Design

- The **Entity-Relationship (E-R)** model is one of the most widely-used conceptual tools (the other one is UML – see 7.9.2 for a quick overview)

- In the E-R model, a DB is modeled as:
  - a collection of **entities**
  - **relationships** among entities.

**Entity** = “object” or “thing” that exists in the enterprise and is distinguishable from other objects.

Examples: specific person, company, place (country, city, county, etc.), event, plant, product, employee
Conceptual Design with the E-R model

- Entities have **attributes**
  - Example: people have *names* and *addresses*

- An **entity set** is a set of entities of the same type that share the same properties.
  - Examples: the set of all persons, companies, trees, holidays
**Entity Sets** **instructor** and **student**

**instructor**

<table>
<thead>
<tr>
<th>instructor_ID</th>
<th>instructor_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766</td>
<td>Crick</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
</tr>
</tbody>
</table>

**student**

<table>
<thead>
<tr>
<th>student-ID</th>
<th>student_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>98988</td>
<td>Tanaka</td>
</tr>
<tr>
<td>12345</td>
<td>Shankar</td>
</tr>
<tr>
<td>00128</td>
<td>Zhang</td>
</tr>
<tr>
<td>76543</td>
<td>Brown</td>
</tr>
<tr>
<td>76653</td>
<td>Aoi</td>
</tr>
<tr>
<td>23121</td>
<td>Chavez</td>
</tr>
<tr>
<td>44553</td>
<td>Peltier</td>
</tr>
</tbody>
</table>
Conceptual Design with the E-R model

Entity sets are not necessarily disjoint, nor covering:

Examples:

- a person can be both a customer and an employee of the bank
- a person can be neither employee nor customer (resist the temptation of representing EVERYTHING, we have to draw the line somewhere)
Relationships

- A relationship is an association among several entities.

Example:
- 44553 (Peltier) \textit{advisor} 22222 (Einstein) \textit{student} entity relationship set \textit{instructor} entity

- A relationship set is a mathematical relation among \(n \geq 2\) entities, each taken from entity sets.

\[\{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}\]

where \((e_1, e_2, \ldots, e_n)\) is a relationship.

Example: \((44553,22222) \in \text{advisor}\)

Subset of the full cartesian product
\[E_1 \times E_2 \times \ldots \times E_n\]
For a **binary** relationship such as this, a relationship set (instance) is simply “a bunch of lines”
Let’s define a ternary relationship `proj_guide` by including the entity `research project` (attributes: ID and name)
A relationship set (instance) is now simply “a bunch of ____________” (fill in the blank)
An attribute can also be property of a relationship set. Example: the *advisor* relationship set may have the attribute *date* which tracks when the student started being associated with the advisor:

```
<table>
<thead>
<tr>
<th>student</th>
<th></th>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766</td>
<td>Crick</td>
<td>98988</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>12345</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>00128</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>76543</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
<td>76653</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>23121</td>
</tr>
</tbody>
</table>
```

QUIZ: Relationship Sets

Instead of the ternary relation defined above, can we capture the project as another attribute that is the property of the binary relation?
Degree of a Relationship Set

- **Binary relationship**
  - involve two entity sets (or degree two).
  - most relationship sets in a DB are binary.

- **Ternary**, quaternary, …, n-ary (“arity” of a relation)

- Relationships between more than two entity sets are rare.

- Most relationships are binary … or they can be **decomposed** into several binary rel. (more on this later)
Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  - Example:
    
    
    \[
    \begin{align*}
    instructor &= (ID, name, street, city, salary) \\
    course &= (course_id, title, credits)
    \end{align*}
    \]
  
  - Domain – the set of permitted values for each attribute
Attributes

Attribute types:

- **Simple** and **composite** attributes.
- **Single-valued** and **multivalued** attributes
  - Example: multivalued attribute: *phone_numbers*
- **Derived** attributes
  - Can be computed from other attributes
  - Example: age can be derived from *date_of_birth*
Composite Attributes

- Composite attributes:
  - name
    - first_name
    - middle_initial
    - last_name
  - address
    - street
    - city
    - state
    - postal_code

- Component attributes:
  - street_number
  - street_name
  - apartment_number
7.3.1 Mapping 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:
  - One to one
  - One to many
  - Many to one
  - Many to many
Mapping Cardinalities

One to one  

One to many  

Note: Some elements in $A$ and $B$ may not be mapped to any elements in the other set
**Mapping Cardinalities**

(a) Many to one

(b) Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set!
Representing CONSTRAINTS

7.3.2 Participation Constraints

- Partial participation
- Total participation
QUIZ: Participation

For each relationship, state if either sets participates partially or totally
7.3.3 Keys

- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.

- A **candidate key** of an entity set is a **minimal** super key
  - *ID* is candidate key of *instructor*
  - *course_id* is candidate key of *course*

- Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key (PK)**.
Keys for Relationship Sets

- The combination of PKs of the participating entity sets forms a super key of a relationship set.
  - \((s_{id}, i_{id})\) is the super key of advisor
  - NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.
    - Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot assume a relationship for each meeting. (We can use a multivalued attribute though.)

- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys

- Need to consider semantics of relationship set in selecting the \(PK\) in case of more than one candidate key.
QUIZ: What are the 3 types of constraints can we represent in the ER model? Give examples of each.
QUIZ: What are the 3 types of constraints can we represent in the ER model? Give examples of each.

Mapping Cardinality c.
Participation c.
Keys
Suppose we have decided to include these entity sets:

- **instructor**, with attributes: ID, name, dept_name, salary
- **department**, with attributes: dept_name, building, budget

and a relationship

- **inst_dept** relating **instructor** and **department**

Problem: Attribute `dept_name` in entity `instructor` is redundant since there is an explicit relationship `inst_dept` which relates instructors to departments

- The attribute replicates information present in the relationship, and should be removed from **instructor**
- BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see.
- HOWEVER: Are instructors associated with only one department? (see discussion on p.273)
7.5 E-R Diagrams

- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes
Entity With Composite, Multivalued, and Derived Attributes

instructor

<table>
<thead>
<tr>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
<tr>
<td>middle_initial</td>
</tr>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>street</td>
</tr>
<tr>
<td>street_number</td>
</tr>
<tr>
<td>street_name</td>
</tr>
<tr>
<td>apt_number</td>
</tr>
<tr>
<td>city</td>
</tr>
<tr>
<td>state</td>
</tr>
<tr>
<td>zip</td>
</tr>
<tr>
<td>{ phone_number }</td>
</tr>
<tr>
<td>date_of_birth</td>
</tr>
<tr>
<td>age ( )</td>
</tr>
</tbody>
</table>

advisor

student

<table>
<thead>
<tr>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>tot_cred</td>
</tr>
</tbody>
</table>
Relationship Sets with Attributes

```
<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>
```

- **Date**
QUIZ: Attributes

Make date a composite attribute: mm, dd, yy
QUIZ: Attributes

Make *date* a multivalued attribute
Make *date* a derived attribute
QUIZ: Attributes

Draw the E-R diagram for two entity sets, car and insurance company, and the relationship set insured by between them.

- **car** has the following attributes: make, model, color, VIN, owner (with a name and address), date of purchase (with year, month, day), and length of ownership.

- **Insurance company** has the following attributes: name, subsidiaries (can be several of them), tax ID, address (with city, street, number and apartment).

- **An insurance** relationship has a start date (with year, month, day), and a monthly payment.
solution
Read pp.259-275 of text

Solve Exercises 7.14 and 7.15
Roles

The entity sets of a relationship need not be distinct

- Each occurrence of an entity set plays a “role” in the relationship

In the example below, the labels “course_id” and “prereq_id” are called roles.
Cardinality Constraints

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.

Examples
One-to-One Relationship between an instructor and a student

- an instructor is associated with at most one student
- a student is associated with at most one instructor
One-to-Many Relationship

- an instructor is associated with several (possibly 0) students
- a student is associated with at most one instructor
  - Don’t confuse with the FK arrow, which means one-and-only-one!

```
  instructor
   ID
   name
   salary

  advisor

  student
   ID
   name
   tot_cred
```
Many-to-One Relationships

- an instructor is associated with at most one student
- and a student is associated with several (possibly 0) instructors
Many-to-Many Relationship

- an instructor is associated with several (possibly 0) students
- a student is associated with several (possibly 0) instructors
Participation of an Entity Set in a Relationship Set

Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set

- E.g.: participation of *section* in *sec_course* is total, every *section* must have an associated course

Partial participation: some entities may not participate in any relationship in the relationship set

- E.g.: participation of *course* in *section* is partial, a course does not necessarily have to have any sections
7.1 Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Each payment is for a particular period of time, and has an associated due date, and the date when the payment was received.

Draw the entity sets first.
7.1 Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Each payment is for a particular period of time, and has an associated due date, and the date when the payment was received.

Include the relationship sets, w/appropriate mapping cardinality and participation constr.
Cardinality limits can also express participation constraints:

```
<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>
```

Relation: advisor

- From instructor to advisor: 0..*
- From advisor to student: 1..1
Ternary Relationship

We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint

- E.g., an arrow from \textit{proj\_guide} to \textit{instructor} indicates each student has at most one guide for a project
If there is more than one arrow, there are two ways of defining the meaning.

- E.g., a ternary relationship $R$ between $A$, $B$ and $C$ with arrows to $B$ and $C$ could mean
  1. each $A$ entity is associated with a unique entity from $B$ and $C$ or
  2. each pair of entities from $(A, B)$ is associated with a unique $C$ entity, and each pair $(A, C)$ is associated with a unique $B$

- Each alternative has been used in different formalisms

To avoid confusion we outlaw more than one arrow
Construct another E-R diagram for the car insurance company; this time, the DB needs to capture not only what cars were involved in each accident, but also who the driver was. Use the name *participated* for the new ternary relationship set.
7.5.6 Weak Entity Sets

An entity set that **does not have a primary key** is referred to as a **weak entity set**.

Needs an **identifying entity set**

- It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
- **Identifying relationship** depicted using a double diamond

![Diagram](image)
Weak Entity Sets

The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set’s discriminator.
Weak Entity Sets

The PK of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.

- If course_id were explicitly stored, section could be made a strong entity, but then the relationship between section and course would be duplicated by an implicit relationship defined by the attribute course_id common to course and section.
Another example of a Weak Entity Set

The cardinality constraints are represented differently b/c there is no standard for E-R diagram notation (See Section 7.9.1)

Source: http://en.wikipedia.org/wiki/Weak_entity
More reasons for using weak entities

7.19 We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Why, then, do we have weak entity sets?

**Answer:** We have weak entities for several reasons:

- We want to avoid the data duplication and consequent possible inconsistencies caused by duplicating the key of the strong entity.
- Weak entities reflect the logical structure of an entity being dependent on another entity.
- Weak entities can be deleted automatically when their strong entity is deleted.
- Weak entities can be stored physically with their strong entities.
- Ease of access: Once the parent key is known, related dependent entities can be accessed quicker via their partial key (discriminator). More on this at indexes – Ch.11.
7.5.7 E-R diagram for the University Enterprise
This is the end of the material required on the midterm.

Individual work (review) for next time:

- Read pp.272-284 of text
- Draw the complete ER diagrams for Practice Exercises 7.1 and 7.2 (including constraints!)