LECTURE 3
ENTITY-RELATIONSHIP (E-R) MODEL AND DATABASE DESIGN
E-R MODEL & DB DESIGN: OUTLINE

- Database Design Process

- Entity-Relationship (E-R) Model — Data Structure Types
  - Entity
  - Attribute
  - Entity Generalization/Specialization
  - Relationship

- Entity-Relationship (E-R) Model — Constraints
  - Attribute — Domain, Key
  - Entity Generalization/Specialization — Coverage
  - Relationship — Cardinality, Participation, Exclusion

Analyzing Application Requirements / Making Design Choices

Reduction of E-R Schemas to Relational Schemas
EXERCISE 1: UNIVERSITY APPLICATION—E-R DIAGRAM

**Student**
- studentId
- name
  - {major}

**Department**
- code
- name

**Course**
- courseId
- name
- prerequisite
- section
- semester
- year

**Offering**
- course
- prerequisite
- grade

**Staff**
- hkid
- name
- officeNumber
- title

**Instructor**
- title

**TA**

**Department**
- Offers

**Course**
- HasPrerequisite
- EnrollsIn

**Staff**
- Appoints
- AssignedTo

**Instructor**

**TA**

**Student**

**Offering**

**Course**

**Department**

**Staff**

**Instructor**

**TA**
A **constraint** is a logical restriction or property of data that for any set of data values:

- we can determine whether the constraint is **true** or **false**;
- we expect the constraint to be **always true**;
- we can **enforce** the constraint.

### Examples:

**on attributes**
- salary is between $0 and $100,000

**on relationships**
- every employee works in **at most** one department
- not every employee is a manager

Constraints add **additional semantics (meaning)** to data (so as to more accurately reflect the application requirements).
ATTRIBUTE CONSTRAINTS: DOMAIN

A domain constraint restricts an attribute to have only certain values.

<table>
<thead>
<tr>
<th>Employee</th>
<th>set of positive integers from 0 to 9999</th>
</tr>
</thead>
<tbody>
<tr>
<td>empId</td>
<td>set of character strings of length 20 or less</td>
</tr>
<tr>
<td>name</td>
<td>set of positive decimal numbers</td>
</tr>
<tr>
<td>address</td>
<td></td>
</tr>
<tr>
<td>hkid</td>
<td></td>
</tr>
<tr>
<td>salary</td>
<td></td>
</tr>
<tr>
<td>jobType</td>
<td></td>
</tr>
</tbody>
</table>

A domain constraint can be specified as a type for the attribute and/or a logical predicate that restricts the values.
ATTRIBUTE CONSTRAINTS: KEY

- If the values of some attributes uniquely identify an entity instance, then they are a key for the entity.

- A candidate key is a minimal set of attributes (i.e., all attributes are needed) that uniquely identifies an entity instance.
  - An entity may have more than one candidate key.

- One candidate key is selected by the database designer to be the primary key.
  - This has enforcement implications for implementation.
    - primary key $\implies$ uniqueness is automatically enforced by a DBMS
    - other candidate keys $\implies$ uniqueness is not automatically enforced by a DBMS

- A candidate/primary key can be composed of a set of attributes $\implies$ composite key.

<table>
<thead>
<tr>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>empId</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>hkid</td>
</tr>
<tr>
<td>salary</td>
</tr>
<tr>
<td>jobType</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EnrollIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>studentId</td>
</tr>
<tr>
<td>courseCode</td>
</tr>
<tr>
<td>grade</td>
</tr>
</tbody>
</table>
**EXERCISE 1: UNIVERSITY APPLICATION—KEYS OF ENTITY TYPES**

- For each student we store the student id, name and majors.
- For each department we store a unique code and name.
- For each course we store a unique course id, name, department and prerequisites.
- For each offering of a course, we store the section, semester and year.
- Each student must enroll in one to five course offerings.
- Each course offering can enroll zero to sixty students.
- For each course offering that a student takes we store the grade.
- Each course offering’s teaching team has one or more staff, who is either an instructor or a TA.
- For each staff assigned to a course offering’s teaching team we store the hkid, name, department and office number.
- For each instructor we store their academic title (e.g., professor).

<table>
<thead>
<tr>
<th>Student</th>
<th>Department</th>
<th>Course</th>
<th>Offering</th>
<th>Staff</th>
</tr>
</thead>
<tbody>
<tr>
<td>studentId name {major}</td>
<td>code name</td>
<td>courseId name</td>
<td>section semester year</td>
<td>hkid name officeNumber</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instructor</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td></td>
</tr>
</tbody>
</table>
**Strong entity**: An entity that *has* a primary key.

**Weak entity**: An entity that *does not have* a primary key.

- A weak entity **must** be associated with a strong entity, called the *identifying entity*, to be meaningful.
  - A weak entity **depends** on its identifying entity for its existence.

- The relationship associating the weak entity to the strong entity is called the *identifying relationship* (shown as a solid line).

- A *discriminator*, *if present*, uniquely identifies a weak entity instance within its identifying relationship.
What kind of entity is Offering?

⟹ Weak entity dependent on Course.

Is there a discriminator for Offering?

⟹ Yes — section, semester, year.
ENTITY GENERALIZATION CONSTRAINTS: COVERAGE

Disjointness

(a) overlapping

A superclass instance can relate to more than one subclass.
E.g., a given company can be both a customer and a supplier at the same time.

(b) disjoint

A superclass instance can relate to at most one subclass.
E.g., a given employee can be either a secretary or an engineer, but not both at the same time.
Completeness

(a) partial
A superclass instance does not need to relate to any of the subclasses.
E.g., a loan does not need to be a mortgage (loan) or a tax (loan) or a car (loan)—there are other kinds of loans.

(b) total
A superclass instance must relate to at least one of the subclasses.
E.g., a given customer must be either a personal or a business customer.
Coverage is specified as one from disjointness (when there is more than one subclass) and one from completeness.
EXERCISE 1: UNIVERSITY APPLICATION—ENTITY GENERALIZATION COVERAGE

- Each course offering’s teaching team has one or more staff, who is either an instructor or a TA.

What should be the disjointness constraint?

⇒ disjoint

What should be the completeness constraint?

⇒ total
Cardinality specifies the maximum number and participation specifies the minimum number of relationship instances in which an entity may participate.

For a given project, how many employees can manage it?
☞ Each project is managed by one and only one employee.

For a given employee, how many projects can he/she manage?
☞ An employee does not have to manage any project, but may manage several (i.e., an unknown number of) projects.
**RELATIONSHIP CONSTRAINTS: CARDINALITY & PARTICIPATION**

**minimum cardinality (min-card) \(\Rightarrow\) participation constraint**

- \(\text{min-card}(E_1,R)\): The *minimum* number of relationship instances in which each entity of \(E_1\) *must* participate in \(R\).

- \(\text{min-card}(E_1,R) = 0 \Rightarrow \text{partial}\) participation

- \(\text{min-card}(E_1,R) > 0 \Rightarrow \text{total}\) participation

**maximum cardinality (max-card) \(\Rightarrow\) cardinality constraint**

- \(\text{max-card}(E_1,R)\): The *maximum* number of relationship instances in which each entity of \(E_1\) *may* participate in \(R\).
An employee does not have to manage a project but can manage several projects. Every project must be managed by an employee and at most one employee.
RELATIONSHIP CONSTRAINTS:
EXAMPLE CARDINALITY & PARTICIPATION

one-to-one (1:1) relationship

Employee

AssignedTo

Office

- John
- James
- Alan

- Room 203
- Room 251
- Room 222
- Room 242

information engineering (crow foot) notation

RELATIONSHIP CONSTRAINTS:
EXAMPLE CARDINALITY & PARTICIPATION

one-to-one (1:1) relationship

Employee

AssignedTo

Office

- John
- James
- Alan

- Room 203
- Room 251
- Room 222
- Room 242

information engineering (crow foot) notation
one-to-many (1:N) relationship

**Relationship Constraints:**

**Example Cardinality & Participation**

- **min-card:** minimum cardinality
- **max-card:** maximum cardinality

**Cardinalities:**
- `min-card (1,N) max-card` for total participation
- `min-card (0,1) max-card` for partial participation

**Diagram:**

- **Employee:** John, James, Alan, Ed, Bill, Larry
- **Department:** Sales, Marketing
- **WorksIn:**
  - Total participation: John, Alan, Bill, Larry
  - Partial participation: James, Ed
many-to-many (N:M) relationship

Example Cardinality & Participation

Employee: John, Alan, Bill, Ed, Larry
Project: E-commerce, Accounting, Stock control, Web store

Relationship Constraints:

- Employee: min-card 0, max-card N
- Project: min-card 0, max-card N

Partial participation in both directions.
A weak entity may be related to more than one strong entity but may depend on only some of these for its existence.

Only some of the strong entities are identifying entities.

How to show which entities are the identifying entities (i.e., which are the identifying relationships)?

The participation of a weak entity in an identifying relationship is always total.

A solid relationship line indicates the identifying relationship(s).
EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

- For each course we store a unique course id, name, department and prerequisites.

What should be the cardinality constraint (max-card) for Department?
- \( \Rightarrow \text{many} \) (A department can offer many courses—domain knowledge.)

What should be the participation constraint (min-card) for Department?
- \( \Rightarrow \text{unknown} \) (Could be partial or total; need to verify with client. Leave unspecified.)

What should be the cardinality constraint (max-card) for Course?
- \( \Rightarrow \text{unknown} \) (Could be 1 or N; need to verify with client. Leave unspecified.)

What should be the participation constraint (min-card) for Course?
- \( \Rightarrow \text{total} \) (Every course must be offered by some department—domain knowledge.)
EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

- For each course we store a unique course id, name, department and prerequisites.

What should be the cardinality constraints?

\[ \rightarrow \text{Course (prerequisite) many} \] (A course can be a prerequisite for several courses.)

\[ \rightarrow \text{Course (course) many} \] (A course can have several prerequisites.)

What should be the participation constraints?

\[ \rightarrow \text{Course (prerequisite) partial} \] (A course does not have to be a prerequisite.)

\[ \rightarrow \text{Course (course) partial} \] (A course can have no prerequisites.)
**EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION**

- For each offering of a course we store the section, semester and year.

What should be the **cardinality constraint** (max-card) for Offering?

⇒ 1 (Every offering is for at most one course—domain knowledge.)

What should be the **participation constraint** (min-card) for Offering?

⇒ total (Every offering must be for some course—domain knowledge.)

What about for **Course**?

⇒ (?, many) min-card most likely 0, but need to verify with client. Leave unspecified.
EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

- Each student must enroll in **one to five** course offerings.
- Each course offering can enroll **zero to sixty** students.

Is Offering dependent on Student?  
⇒ No.

What should be the cardinality constraint (max-card) for Student?  
⇒ 5 (A student can enroll in at most 5 course offerings.)

What should be the participation constraint (min-card) for Student?  
⇒ total (A student must enroll in at least 1 course offering.)

What about for Offering?  
⇒ (0, 60)

Does the participation constraint for Student make sense?

Is a student required to enroll in an offering as soon as the student’s record is created?  
No! (domain knowledge)
EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

- Each course offering’s teaching team has one or more staff, who is either an instructor or a TA

Is Offering dependent on Staff?
⇒ No.

What should be the cardinality constraint (max-card) for Offering?
⇒ many (An offering can have several staff assigned to it.)

What should be the participation constraint (min-card) for Offering?
⇒ total (An offering has at least one staff assigned to it.)

What about for Staff?
⇒ (?,?,many) min-card most likely 0, but need to verify with client. Leave unspecified.

Does the participation constraint for Offering make sense?

Is an offering required to have a staff assigned to it?

Need to verify with client!
EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

- For each staff assigned to a course offering’s teaching team we store the hkid, name, department and office number.

What should be the cardinality constraint (max-card) for Staff?
⇒ 1 (For each staff … we store the … department ….)
What should be the participation constraint (min-card) for Staff?
⇒ total (Every staff must be appointed in some department—domain knowledge.)
What should be the cardinality constraint (max-card) for Department?
⇒ many (A department can appoint several staff—domain knowledge.)
What should be the participation constraint (min-card) for Department?
⇒ unknown (Could be partial or total; need to verify with client. Leave unspecified.)
EXERCISE 1: UNIVERSITY APPLICATION—E-R DIAGRAM

Student
- studentId
- name
  (major)

Department
- code
- name

Course
- courseId
- name
- prerequisite

Offering
- section
- semester
- year

Staff
- hkid
- name
- officeNumber

Instructor
- title

TA

Staff
- has
- appointed

Offering
- has
- enrollsIn

Student
- enrollsIn

Department
- offers

Course
- hasPrerequisite

Instructor
- teaches

TA
- teaches

EXERCISE 1: UNIVERSITY APPLICATION—E-R DIAGRAM
An exclusion (XOR) constraint specifies that at most one entity instance, among several entity types, can participate in a relationship with a single “root” entity.

Example: A task can be related to either an internal project or an external project, but not both.
An exclusion (XOR) constraint specifies that at most one entity instance, among several entity types, can participate in a relationship with a single “root” entity.

Example: A task can be related to either an internal project or an external project, but not both.
E-R MODEL & DB DESIGN: OUTLINE

✓ Database Design Process

✓ Entity-Relationship (E-R) Model — Data Structure Types
  – Entity
  – Attribute
  – Generalization/Specialization
  – Relationship

✓ Entity-Relationship (E-R) Model — Constraints
  – Attribute — Domain, Key
  – Generalization/Specialization — Coverage
  – Relationship — Cardinality, Participation, Exclusion

→ Analyzing Application Requirements / Making Design Choices

Reduction of E-R Schemas to Relational Schemas
ANALYZING APPLICATION REQUIREMENTS

1. Identify entities
   - What are the major concepts about which data needs to be permanently stored?
   - Focus on the “big picture”, not the details.
     ➢ E.g., student, course not name, address, email, description, credits, etc.

2. Identify relationships between entities
   - How are the major concepts related? How do they interact?
   - What interactions need to be permanently stored.
     ➢ E.g., students enroll in courses not students browse courses

3. Identify properties of entities and relationships
   - For each entity and relationship, what information needs to be permanently stored.
DESIGN CHOICE: ENTITY VERSUS ATTRIBUTE

entity: When several properties can be associated with the concept.

attribute: When the concept has a simple atomic structure or no property of interest.

Office as attribute

Office as entity
Where to place salary?

Relationship attributes are usually needed only for many to many relationships!
(But can also be used in one to one and one to many relationships.)
strong entity: When the concept can be uniquely identified in the application domain (i.e., it has a key).

weak entity: When the concept has no unique identifier.

Suppose an account must be associated with exactly one branch and two different branches can have accounts with the same number.

Should Account be a strong or weak entity?
DESIGN CHOICE: ENTITY VERSUS RELATIONSHIP

**entity:** When the concept represents something distinct in the application domain with several properties.

**relationship:** When the concept is not a distinct application domain concept and/or has no property of interest.

**Account as entity**

![Diagram showing Entity Relationship Diagram with Customer, Account, and Branch entities and relationship with Holds and IsAt attributes.]

**Account as relationship**

![Diagram showing Entity Relationship Diagram with Customer, Account, and Branch entities and relationship without Holds and IsAt attributes.]

What if you want to have several accounts for a customer at the same branch?
We want to represent the fact that James has two accounts at the same branch.

An entity can be related to another entity at most once by a given relationship.
We need to use an entity for Account!

There can be only one relationship instance of a given relationship type between the same two entity instances.
E-R MODEL & DATABASE DESIGN

EXERCISE 2

Upload your completed exercise worksheet to Canvas by 11 p.m.