CS 1555
www.cs.pitt.edu/~nlf4/cs1555/
ER-modeling
Have a problem, but how do we best address is data management needs?

- How many tables do we need?
- What attributes do we create in which tables?
- What relationships do we establish between tables?

ER-Modeling can help guide us through this process
Designing a database application

- Overall design will work along two paths
  - Functional design
  - Conceptual design
What does the database application need to do?
What does the data need to be used for?
Will start off as a general sketch of transaction
Will lead to finalized transaction and an application
How should we structure the database in order to do what is needed?

- This is where ER-Modeling will help
- Will result in logical schema for the database
  - And possibly a physical schema
ER-modeling basics

● We will use it to drive conceptual design

● Two main primitives:
  ○ Entities
    ■ With physical existence
      ● Alice, Bob, Alice's house
    ■ Or conceptual existence
      ● Bob's bank account, CS1555
  ○ Relationships
    ■ Alice married Bob
    ■ Alice took CS1555
Attributes

- Properties of entities:
  - The color of Alice's car
  - Bob's age

- Or of relationships:
  - Alice married Bob on July 4
  - Alice was hired at her job 6 years ago
  - Alice took CS1555 in Fall 2008
More on attributes

● What is their structure?
  ○ Simple, atomic value
  ○ Composite
    ■ E.g., street address

● What is their value?
  ○ Single-value
  ○ Multi-value
    ■ E.g., list of degrees
  ○ Derived-value
  ○ NULL
Entity types

- All similar entities (those with the same attributes) are grouped into the same *entity type*
- E.g., FACULTY
  - Name(FN,LM,MI), DoB, SSN, {Degree}, Rank
    - FN: String(15)
    - LN: String(15)
    - MI: String(1)
    - SSN: String(9)
    - DoB: DD/MM/YYYY
    - Degree: {BS,MS,PhD}
    - Rank: {Lecturer, Assistant, Associate, Full}
Relationship types

● Describe how entities of different types can relate to one another
  ○ E.g., ENROLLS: <Student, Course>
● Degree is the number of participating entity types
  ○ 2 types: binary relationship
  ○ 3 types: ternary
    ■ E.g., suppliers supply parts for a project
  ○ ...
  ○ n types: n-ary
● Recursive relationship involve the same entity type in multiple different roles:
  ○ MENTORS: <mentor-faculty, mentored-faculty>
Relationship cardinality ratios

- **1:1**
  - One-to-one

- **1:N**
  - One-to-many

- **N:1**
  - Many-to-one

- **N:M**
  - Many-to-many
Participation

● Total participation
  ○ Every entity of a given type *must* have a certain relationship with some entity of another type
    ■ E.g., every employee belongs to a department

● Partial participation
  ○ Some entities of a given type have a certain relationship with some entity of another type
    ■ E.g., some employees manage departments
Strong and weak entities

- **Strong or ordinary entities**
  - Have independent existence in the mini-world
  - They are part of the core of the application

- **Weak entities**
  - They are dependent on another entity
  - *Identifying or owner* entity type is the specific entity on which the weak entity depends
  - No key attribute; are distinguishable through an identifying relationship and a *discriminator* or *partial key*
  - Identifying relationship is always total participation
  - It may be represented as multi-value, composite attribute of owner
ER diagrams

- Entity
- Weak Entity
- Relationship
- Identifying Relationship
ER diagrams

- Attribute
- Key Attribute
- Multivalued Attribute
ER diagrams

Composite Attribute

Derived Attribute
ER diagrams

- Total Participation of $E_2$ in $R$
- Cardinality Ratio 1: N for $E_1:E_2$ in $R$
- Structural Constraint (min, max) on Participation of $E$ in $R$
Library example case study

- Organized into sections
  - E.g., art, childrens, computing, science, etc.
  - Each section has name and a number and is headed by a head librarian
- Each book title belongs to a section and has a title, authors, an ISBN, a call number, a year, and a publisher
- For each copy of the book, keep track the current borrower, the due date, and the librarian who charged it out
- Members have a membership number, a driver’s license, an address, a phone number, and birthday
- Members can have up to 5 borrowed books and can put a hold request on a book
- Librarians have a name, an ssn, an address, and a phone
A general rule of thumb

- nouns → entity types/sets
- verbs → relationship types
Entities

- **TITLE**: CallNumber, Name, ISBN, Author{(Name(Fname, MI, Lname),Order)}, Year, Publisher
- **MEMBER**: MemNo, DriverLic(State,No), Name(Fname, MI, Lname), Address, PhoneNumber
- **BOOK**: BookID, Edition
- **LIBRARIAN**: SSN, Name, Address, Salary, DoB
- **SECTION**: SectNo, Name
- Assume the additional requirement that all the dependents of each librarian are stored in the DB:
  - **DEPENDENT**: Name(Fname, MI, Lname), DoB, Kinship
Relationships

- COPY: <TITLE, BOOK>
- BELONGS: <BOOK, SECTION>
- HOLD: <MEMBER, TITLE>
- BORROW: <MEMBER, BOOK>
- CHECKS: <LIBRARIAN, BOOK>
- MANAGES: <LIBRARIAN, SECTION>
- WORKS: <LIBRARIAN, SECTION>
- DEPENDS: <LIBRARIAN, DEPENDENT>
- SUPERVISES: <supervisor-LIBRARIAN, supervisee-LIBRARIAN>
Library
ER Diagram
Another ER Model Example

Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.
How do we build tables off of this example??

Figure 9.2
Result of mapping the COMPANY ER schema into a relational database schema.
Enhanced ER modeling (EER modeling)

- The EER model introduced the concepts of superclass and subclass entity types in the ER model
- MEMBER (superclass):
  - LIFE_MEMBER, REGULAR_MEMBER, and SEASON_MEMBER (subclasses)
- LIBRARIAN (superclass):
  - HEAD_LIBRARIAN, SALARY_LIBRARIAN, and HOURLY_LIBRARIANS (subclasses)
Why EER?

- How does this help us set up a database schema?
Can also help clarify relationships

- If only salary-librarians can belong to the librarian guild, then this can be expressed as:
  - BelongTo:<SALARY_LIBRARIAN, LIB_GUILD>

- Instead of:
  - BelongTo:<LIBRARIAN, LIB_GUILD>
Specialization, generalization, and inheritance

- **Specialization**
  - Identifying subclasses, and their distinguishing characteristics (attributes & relationships)
  - Top-down design

- **Generalization**
  - Aggregate entities to a superclass entity type by identifying their common characteristics
  - Bottom-up design

- **Inheritance**
  - IS_A (instance) relationship that supports attribute inheritance and relationship participation
  - Single inheritance results in a hierarchy
  - Multiple inheritance results in a lattice
    - E.g., EMPLOYEE → STUDENT-ASSISTANT ← STUDENT
Subclass definition

- The entities for each class can be user-defined or specified with a condition on attributes of from the superclass
  - In a predicate-defined subclass, we use a selection condition on one or more attributes to define the entities of the subclass
  - Attribute-defined specializations occur when the same attribute of the superclass is used to determine membership in all subclasses (e.g., MembershipType)
Inclusion constraint classifications

● Disjoint constraints
  ○ The subclasses of a superclass are disjoint.
    ■ This means that an entity can be a member of only one subclass

● Overlapping constraints
  ○ Specify that the subclasses are overlapping and an entity may be a member of more than one subclass.
Completeness constraints

● Total specialization
  ○ Specifies that every entity in the superclass must be a member of some of its subclasses
  ○ E.g., a librarian must belong to one of the subclasses of LIBRARIAN

● Partial specialization
  ○ Specifies that an entity may not belong to any subclass
  ○ E.g., an honorary member may not belong to any of the specializations (subclasses) of MEMBER

● Superclass via generalization is always total
Union types (or categories)

- Collection of entities of distinct entity types
  - A vehicle owner could be a person, a bank, or a company
- Multiple Inheritance with superclasses of different types
- Category OWNER is a subclass of the set union of the entity types: PERSON, BANK, COMPANY
- An instance in category must exists only in one of the superclasses
- Category can be
  - Total
  - Partial (with predicate definition)