CMSC424: Database Design
Entity-Relationship Model

Instructor: Amol Deshpande
amol@cs.umd.edu
Outline

- Database Design Process
- Entity-relationship Model (E/R model)
- Converting from E/R to Relational
- Extra slides
Database Design Process

- **Why?**
  - Difficult to directly create schemas for complex domains
  - Need significant back-and-forth between the developer and the users

- **Common Steps:**
  - Initial design: Characterize the data needs of the users, including functional requirements (what types of queries/transactions)
  - Choose a data model appropriate for the data needs
  - Translate the requirements into a “conceptual schema”
  - Logical Design Step: Convert to the logical schema, typically relational
  - Physical Design Steps: Decide physical layout of the database

- **Normalization (covered later) also deals with this issue**
Outline

- Database Design Process
- Entity-relationship Model (E/R model)
- Converting from E/R to Relational
- Extra slides
Entity-Relationship Model

- Conceptual schema often done in the E/R Model
- Why?
  - Why not just use the relational model directly?
  - Relational model too impoverished
    - Hard to understand what’s going on
    - No distinction between different types of entities or relationships
      - Everything is a table
    - Too much detail
- E/R models have an associated diagrammatic representation
  - Easier to work with in the initial design phases
- At the end: easy to convert to a relational schema (almost mechanical)
- Key entities and "relationships" between them, all mixed up.
- Attributes appearing multiple times
- Complicated foreign keys
7.5.7 E-R diagram for the University Enterprise

In Figure 7.15, we show an E-R diagram that corresponds to the university enterprise that we have been using thus far in the text. This E-R diagram is equivalent to the textual description of the university E-R model that we saw in Section 7.4, but with several additional constraints, and section now being a weak entity.

In our university database, we have a constraint that each instructor must have exactly one associated department. As a result, there is a double line in Figure 7.15 between instructor and inst_dept, indicating total participation of instructor in inst_dept; that is, each instructor must have exactly one associated department. Furthermore, there is an arrow from inst_dept to department, indicating that each instructor can have at most one associated department.

**Entities**

- course
  - course_id
  - title
  - credits
- inst_dept
- inst_dept
- instructor
  - ID
  - name
  - salary
- section
  - sec_id
  - semester
  - year
- sec_course
  - course_id
  - prereq_id
- sec_time_slot
  - time_slot_id
    - day
    - start_time
    - end_time
- sec_class
- classroom
  - building
  - room_number
  - capacity
- course_dept
- department
  - dept_name
  - building
  - budget
- student
  - ID
  - name
  - tot_cred
- student
- advisor
- teaches
- takes
- grade

**Relationships**

- Instructor teaches Section
- Instructor takes Section
- Instructor advises Student
- Student takes Section
- Section is taught in Classroom
- Section has Time Slot
- Classroom is in Building
- Course is in Department
- Department has Budget
- Student can take at most one Section
- Instructor must have exactly one Department

---

VS.

*Relationships between them*
Two key concepts

- **Entities:**
  - An object that *exists* and is *distinguishable* from other objects
  - Examples: Bob Smith, BofA, CMSC424
  - Have *attributes* (people have names and addresses)
  - Form *entity sets* with other entities of the same type that share the same properties
    - Set of all people, set of all classes
  - Entity sets may overlap
    - Customers and Employees
Entity-Relationship Model

- Two key concepts
  - **Relationships**:
    - Relate 2 or more entities
      - E.g. Bob Smith *has account at* College Park Branch
    - Form *relationship sets* with other relationships of the same type that share the same properties
      - Customers *have accounts at* Branches
    - Can have attributes:
      - *has account at* may have an attribute *start-date*
    - Can involve more than 2 entities
      - Employee *works at* Branch *at* Job
Entities and relationships

Two Entity Sets

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Student</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>
<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>

Advisor Relationship, with and without attributes

instructor

student

3 May 2008
10 June 2007
12 June 2006
6 June 2009
30 June 2007
31 May 2007
4 May 2006
Alternative representation, used in the book in the past
Rest of the class

- Details of the ER Model
  - How to represent various types of constraints/semantic information etc.

- Design issues

- A detailed example
Next: Relationship Cardinalities

- We may know:
  - One student can only have one advisor
  - OR
  - One student can have multiple advisors

- Representing this is important

- Why?
  - Better manipulation of data
    - If former, can store the advisor info in the student table
  - Can enforce such a constraint
    - Application logic will have to do it; NOT GOOD
  - Remember: If not represented in conceptual model, the domain knowledge may be lost
Mapping Cardinalities

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

- **One-to-One**
- **One-to-Many**
- **Many-to-One**
- **Many-to-Many**
Express the number of entities to which another entity can be associated via a relationship set

Most useful in describing binary relationship sets

N-ary relationships?
  - More complicated
  - Details in the book

Figure 7.13 E-R diagram with a ternary relationship.
Next: Types of Attributes

- Simple vs Composite
  - Single value per attribute?

- Single-valued vs Multi-valued
  - E.g. Phone numbers are multi-valued

- Derived
  - If date-of-birth is present, age can be derived
  - Can help in avoiding redundancy, enforcing constraints etc...
Types of Attributes

- **Primary key underlined**: ID
- **Composite**: name, first_name, middle_initial, last_name, address, street, street_number, street_name, apt_number, city, state, zip
- **Multi-valued**: { phone_number }
- **Derived**: date_of_birth, age()
What attributes are needed to represent a relationship completely and uniquely?
- Union of primary keys of the entities involved, and relationship attributes
- \{instructor\_id, date, student\_id\} describes a relationship completely
Is \{instructor\_id, date, student\_id\} a candidate key?

- No. Attribute *date* can be removed from this set without losing key-ness
  - By definition: there can only be one relationship of a given type between two different entities
  - There may be two different relationships between two entities
    - e.g., (instructor is member of a department) and (instructor is “chair” of a department)

- In fact, union of primary keys of associated entities is always a superkey
Is \{\text{instructor-id, student-id}\} a candidate key?

- Depends

- If one-to-one relationship, either \{\text{student_id}\} or \{\text{instructor_id}\} sufficient
  - Since a given student can only have one advisor, she can only participate in one relationship
  - Ditto instructor
- Is \{instructor-id, student-id\} a candidate key?
  - Depends

- If one-to-many relationship (as shown), \{instructor_id\} is a primary key
  - A given instructor can only have one advisee, but a student may have many advisors
  - So the instructor_id is sufficient to uniquely identify a relationship
General rule for binary relationships
- one-to-one: primary key of either entity set
- one-to-many: primary key of the entity set on the many side
- many-to-many: union of primary keys of the associate entity sets

n-ary relationships
- More complicated rules
What have we been doing
  ◦ Try to record all the requirements and constraints

Why?
  ◦ So we can design a faithful and easy-to-use schema

Understanding this is important
  ◦ Rest are details !!
  ◦ That’s what books/manuals are for.
Sometimes a relationship associates an entity set to itself

- Need “roles” to distinguish

<table>
<thead>
<tr>
<th>course</th>
<th>course_id</th>
<th>title</th>
<th>credits</th>
<th>course_id</th>
<th>prereq_id</th>
<th>prereq</th>
</tr>
</thead>
</table>

Recursive Relationships
Weak Entity Sets

- An entity set without enough attributes to have a primary key
- E.g. Section Entity
- Still need to be able to distinguish between weak entities
  - Called “discriminator attributes”: dashed underline
Participation Constraints

- Records the information that any entity in an entity set must participate in at least one relationship of that type.
Specialization/Generalization

Similar to object-oriented programming: allows inheritance etc.

```
<table>
<thead>
<tr>
<th>person</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>address</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>salary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>tot_credits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>secretary</th>
</tr>
</thead>
<tbody>
<tr>
<td>hours_per_week</td>
</tr>
</tbody>
</table>
```
No relationships allowed between relationships

Suppose we want to record evaluations of a student by a guide on a project
Thoughts...

- Nothing about actual data
  - How is it stored?
- No talk about the query languages
  - How do we access the data?
- Semantic vs Syntactic Data Models
  - Remember: E/R Model is used for conceptual modeling
  - Many conceptual models have the same properties
- They are much more about representing the knowledge than about database storage/querying
Basic design principles

- Faithful
  - Must make sense
- Satisfies the application requirements
- Models the requisite domain knowledge
  - If not modeled, lost afterwards
- Avoid redundancy
  - Potential for inconsistencies
- Go for simplicity

Typically an iterative process that goes back and forth
Design Issues

- Entity sets vs attributes
  - Depends on the semantics of the application
  - Consider telephone

- Entity sets vs Relationship sets
  - Consider loan

- N-ary vs binary relationships
  - Possible to avoid n-ary relationships, but there are some cases where it is advantageous to use them

- It is not an exact science!!
Recap

- Entity-relationship Model
  - Intuitive diagram-based representation of domain knowledge, data properties etc...
  - Two key concepts:
    - Entities
    - Relationships
  - We also looked at:
    - Relationship cardinalities
    - Keys
    - Weak entity sets
    - ...
Recap

- Entity-relationship Model
  - No standardized model (as far as I know)
    - You will see different types of symbols/constructs
  - Easy to reason about/understand/construct
  - Not as easy to implement
    - Came after the relational model, so no real implementation was ever done
    - Mainly used in the design phase
Outline

- Entity-relationship Model (E/R model)
- Converting from E/R to Relational
- Extra slides
E/R Diagrams ⇒ Relations

- Convert entity sets into a relational schema with the same set of attributes

```
Customer

cname  ccity  cstreet

Customer_Schema(cname, ccity, cstreet)

Branch

bname  bcity  assets

Branch_Schema(bname, bcity, assets)
```
- Convert relationship sets also into a relational schema.
- Remember: A relationship is completely described by primary keys of associate entities and its own attributes.

Well… Not quite. We can do better. It depends on the relationship cardinality.

Diagram:
- **Account** (acct-no, balance, access-date)
- **Depositor** (cname, acct-no, access-date)
- **Customer** (cname, ccity, cstreet)

**Account_Schema**(acct-no, balance)

**Depositor_Schema**(cname, acct-no, access-date)

**Customer_Schema**(cname, ccity, cstreet)
E/R Diagrams \( \Rightarrow \) Relations

- Say One-to-Many Relationship from Customer to Account
  \( \Rightarrow \) Many accounts per customer

Account Schema: \( \text{Account}_\text{Schema}(\text{acct-no}, \text{balance}, \text{cname}, \text{access-date}) \)

Customer Schema: \( \text{Customer}_\text{Schema}(\text{cname}, \text{ccity}, \text{cstreet}) \)
## E/R Diagrams ➔ Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity Sets</strong></td>
<td></td>
</tr>
<tr>
<td>$E_1$</td>
<td>$E = (a_1, \ldots, a_n)$</td>
</tr>
<tr>
<td>$a_1$</td>
<td></td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>$a_n$</td>
<td></td>
</tr>
</tbody>
</table>

| **Relationship Sets** | 
| $E_1$ | $R = (a_1, b_1, c_1, \ldots, c_n)$ |
| $R$ |  |
| $E_2$ |  |
| $a_1$ | $a_1$: $E_1$’s key |
| $\ldots$ |  |
| $a_n$ |  |
| $c_1$ | $b_1$: $E_2$’s key |
| $\ldots$ |  |
| $c_k$ | $c_1, \ldots, c_k$: attributes of $R$ |
| $b_1$ |  |
| $\ldots$ |  |
| $b_m$ |  |

*Not the whole story for Relationship Sets ...*
### E/R Diagrams \( \rightarrow \) Relations

<table>
<thead>
<tr>
<th>Relationship Cardinality</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="https://via.placeholder.com/150" alt="Diagram" /></td>
</tr>
</tbody>
</table>

#### n:m
- \( R \)
- \( E_1 = (a_1, \ldots, a_n) \)
- \( E_2 = (b_1, \ldots, b_m) \)
- \( R = (a_1, b_1, c_1, \ldots, c_n) \)

#### n:1
- \( R \)
- \( E_1 = (a_1, \ldots, a_n, b_1, c_1, \ldots, c_n) \)
- \( E_2 = (b_1, \ldots, b_m) \)

#### 1:n
- \( R \)
- \( E_1 = (a_1, \ldots, a_n) \)
- \( E_2 = (b_1, \ldots, b_m, a_1, c_1, \ldots, c_n) \)

#### 1:1
- \( R \)
- Treat as n:1 or 1:n
Q. How many tables does this get translated into?

A. 6 (account, branch, customer, loan, depositor, borrower)
### E/R Diagrams & Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Weak Entity Sets</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E&lt;sub&gt;1&lt;/sub&gt;</th>
<th>E&lt;sub&gt;2&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>a&lt;sub&gt;1&lt;/sub&gt;</td>
<td>b&lt;sub&gt;1&lt;/sub&gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>a&lt;sub&gt;n&lt;/sub&gt;</td>
<td>b&lt;sub&gt;m&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

E<sub>1</sub> = (a<sub>1</sub>, ..., a<sub>n</sub>)
E<sub>2</sub> = (a<sub>1</sub>, b<sub>1</sub>, ..., b<sub>m</sub>)
## E/R Diagrams & Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multivalued Attributes</strong></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Employee Diagram" /></td>
<td><strong>Emp</strong> = (ssn, name)</td>
</tr>
<tr>
<td></td>
<td><strong>Emp-Phones</strong> = (ssn, phone)</td>
</tr>
<tr>
<td></td>
<td>ssn  name  phone</td>
</tr>
<tr>
<td>001  Smith</td>
<td>001  4-1234</td>
</tr>
<tr>
<td>...  ...</td>
<td>001  4-5678</td>
</tr>
<tr>
<td></td>
<td><strong>Emp</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Emp-Phones</strong></td>
</tr>
</tbody>
</table>
### E/R Diagrams & Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subclasses</strong></td>
<td></td>
</tr>
</tbody>
</table>

Method 1:

- $E = (a_1, \ldots, a_n)$
- $E_1 = (a_1, b_1, \ldots, b_m)$
- $E_2 = (a_1, c_1, \ldots, c_k)$

Method 2:

- $E_1 = (a_1, \ldots, a_n, b_1, \ldots, b_m)$
- $E_2 = (a_1, \ldots, a_n, c_1, \ldots, c_k)$
Subclasses example:

Method 1:

Account = (acct_no, balance)
SAccount = (acct_no, interest)
CAccount = (acct_no, overdraft)

Method 2:

SAccount = (acct_no, balance, interest)
CAccount = (acct_no, balance, overdraft)

Q: When is method 2 not possible?
A: When subclassing is partial