The Entity-Relationship Model

T. M. Murali

September 23, 2009
Till the Midterm Examination

- Relational Data Models
  - The Entity-Relationship (ER) model
  - The relational model
  - Converting E/R diagram to relational designs.

- At this point, you will know how to
  1. Identify all entities and relationships and describe them using an E/R diagram.
  2. Convert the E/R model to a number of relations in a relational schema.

- Use all these ideas to design your own database application in your project.
Basic Database Terminology

- **Data model**: describes high-level conceptual structuring of data
  - Example: Data is set of student records, each with ID, name, address, and courses
  - Example: Data is a graph where nodes represent proteins and edges represent chemical bonds between proteins

- **Schema** describes how data is to be structured and stored in a database
  - Defined during creation of the database
  - Schemas rarely change

- **Data** is actual “instance” of database
  - Updated continuously
  - Changes rapidly
Why Learn About Database Modelling?

- The way in which data is stored is very important for subsequent access and manipulation by SQL.

- Properties of a good data model:
  - It is easy to write correct and easy to understand queries.
  - Minor changes in the problem domain do not change the schema.
  - Major changes in the problem domain can be handled without too much difficulty.
  - Can support efficient database access.
Purpose of the E/R Model

- The *Entity-Relationship (E/R) model* enables us to draw diagrams of database designs
  - Represent different types of data and how they relate to each other.
- The design is a drawing called the *E/R diagram*.
- When designing E/R diagrams, *forget about relations/tables*; only consider how to model the information you need to represent in your database.
- In two weeks, we will learn how to convert an E/R diagram to a relational schema.
Entity Sets

- An entity is an (abstract) object of some sort.
- An entity set is a collection of similar entities.
- Entities have attributes
  - An attribute is a property of the entities in an entity set
  - In this class, our convention is to use “atomic” attributes (strings, numbers, ...)

T. M. Murali September 23, 2009 CS 4604: E/R model
Entity Sets

- An entity is an (abstract) object of some sort.
- An entity set is a collection of similar entities.
- Entities have attributes
  - An attribute is a property of the entities in an entity set
  - In this class, our convention is to use “atomic” attributes (strings, numbers, ...)
- Analogy between entity sets and classes (in OO languages)
- Analogy between entities and objects.
Entity Sets in an E/R Diagram

- In an E/R diagram
  - A rectangle represents an entity set
  - An oval represents an attribute
  - A line connects an entity set (rectangle) to an attribute (oval)
Examples of Entity Sets

Students
- PID
- Name
- Address

Courses
- Name
- Number
- Classroom
- DeptName
A relationship is a connection between two or more entity sets.

In an E/R diagram,

- a diamond represents a relationship
- a line connects the relationship to each entity set.

Do not confuse “Relationships” with “Relations”.

Examples of Relationships

- **Students Take Courses, Professors Teach Courses, Professors Advise Students**

![Entity-Relationship Diagram]

- Attributes types: strings, numbers, or "enums" (A Professor’s Age could be "old," "much older," or "still alive!").

T. M. Murali  September 23, 2009  CS 4604: E/R model
Examples of Relationships

- **Students Take Courses, Professors Teach Courses, Professors Advise Students**

- Attributes types: strings, numbers, or “enums” (A Professor’s Age could be “old,” “much older,” or “still alive!”).
Instance of an E/R Diagram

- An E/R diagram is a notation for specifying the schema (structure) of a database. It is not an implementation of a database.
- Still useful to think about the *instance* of an E-R diagram: the particular data stored in the database.
Instance of an Entity Set

- For each entity set, the instance stores a specific set of entities.
- Each entity is a \textit{tuple} containing specific values for each attribute.

Example: An instance of the entity set \textit{Students}

- Name: Hermione Grainger, PID: HG, Address: Gryffindor Tower
- Name: Draco Malfoy, PID: DM, Address: Slytherin Tower
- Name: Harry Potter, PID: HP, Address: Gryffindor Tower
- Name: Ron Weasley, PID: RW, Address: Gryffindor Tower
Instance of an Entity Set

- For each entity set, the instance stores a specific set of entities.
- Each entity is a *tuple* containing specific values for each attribute.
- Example: An instance of the entity set *Students*

<table>
<thead>
<tr>
<th>Name</th>
<th>PID</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hermione Grainger</td>
<td>HG</td>
<td>Gryffindor Tower</td>
</tr>
<tr>
<td>Draco Malfoy</td>
<td>DM</td>
<td>Slytherin Tower</td>
</tr>
<tr>
<td>Harry Potter</td>
<td>HP</td>
<td>Gryffindor Tower</td>
</tr>
<tr>
<td>Ron Weasley</td>
<td>RW</td>
<td>Gryffindor Tower</td>
</tr>
</tbody>
</table>
Instance of a Relationship

▶ Example: An instance of the relationship Takes (no DeptName)

<table>
<thead>
<tr>
<th>Student</th>
<th>PID</th>
<th>Address</th>
<th>CourseName</th>
<th>Enrollment</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hermione Grainger</td>
<td>HG</td>
<td>Gryffindor</td>
<td>Potions</td>
<td>∞</td>
<td>A-</td>
</tr>
<tr>
<td>Draco Malfoy</td>
<td>DM</td>
<td>Slytherin</td>
<td>Potions</td>
<td>∞</td>
<td>B</td>
</tr>
<tr>
<td>Harry Potter</td>
<td>HP</td>
<td>Gryffindor</td>
<td>Potions</td>
<td>∞</td>
<td>A</td>
</tr>
<tr>
<td>Ron Weasley</td>
<td>RW</td>
<td>Gryffindor</td>
<td>Potions</td>
<td>∞</td>
<td>C</td>
</tr>
</tbody>
</table>

▶ A relationship $R$ between entity sets $E$ and $F$ relates some entities in $E$ to some entities in $F$.
▶ $R$ is a set of pairs of tuples $(e, f)$ where $e$ is in $E$ and $f$ is in $F$.
  ▶ $R$ need not relate every tuple in $E$ with every tuple in $F$.
  ▶ Relationship set for $R$: all pairs of tuples $(e, f)$ related by $R$.
▶ An instance of $R$ is simply the “concatentation” of the attribute lists for all pairs of tuples $(e, f)$ in the relationship set for $R$.
▶ “Tuples” in $R$ have two components, one from $E$ and one from $F$. 
Instance of a Relationship

Example: An instance of the relationship Takes (no DeptName)

<table>
<thead>
<tr>
<th>Student</th>
<th>PID</th>
<th>Address</th>
<th>CourseName</th>
<th>Enrollment</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hermione Grainger</td>
<td>HG</td>
<td>Gryffindor</td>
<td>Potions</td>
<td>∞</td>
<td>A-</td>
</tr>
<tr>
<td>Draco Malfoy</td>
<td>DM</td>
<td>Slytherin</td>
<td>Potions</td>
<td>∞</td>
<td>B</td>
</tr>
<tr>
<td>Harry Potter</td>
<td>HP</td>
<td>Gryffindor</td>
<td>Potions</td>
<td>∞</td>
<td>A</td>
</tr>
<tr>
<td>Ron Weasley</td>
<td>RW</td>
<td>Gryffindor</td>
<td>Potions</td>
<td>∞</td>
<td>C</td>
</tr>
</tbody>
</table>

A relationship $R$ between entity sets $E$ and $F$ relates some entities in $E$ to some entities in $F$.

$R$ is a set of pairs of tuples $(e, f)$ where $e$ is in $E$ and $f$ is in $F$.

- $R$ need not relate every tuple in $E$ with every tuple in $F$.
- *Relationship set* for $R$: all pairs of tuples $(e, f)$ related by $R$.

An instance of $R$ is simply the “concatentation” of the attribute lists for all pairs of tuples $(e, f)$ in the relationship set for $R$.

“Tuples” in $R$ have two components, one from $E$ and one from $F$.

Question: What is *Grade* an attribute of?
Attributes for a Relationship

- It is useful/essential to attach attributes to relationships.
- Such an attribute is a property of the entity-pairs in the relationship.

What is the meaning of the arrow in the E-R diagram?
Multiplicity in Pictures

Many–many
- Take
- Students
- Classes

Many–one
- Advisor
- Students
- Professors

One–one
- Office
- Rooms
- Professors
Multiplicty of a Binary Relationship

- $R$ is a *many-one relationship* from $E$ to $F$ if each entity in $E$ can be connected to at most one entity in $F$ by $R$.

"at most one" and not "at least one" or "exactly one".

$R$ is *one-one* if it is many-one from $E$ to $F$ and many-one from $F$ to $E$.

Otherwise, $R$ is a *many-many relationship*.

The schema defines the multiplicity of relationships. Don’t use the instances of the schema to determine multiplicity.
Multiplicity of a Binary Relationship

- **R** is a *many-one relationship* from *E* to *F* if each entity in *E* can be connected to at most one entity in *F* by *R*.
  - “at most one” and not “at least one” or “exactly one”.

- **R** is *one-one* if it is many-one from *E* to *F* and many-one from *F* to *E*.

- Otherwise, **R** is a *many-many* relationship.

The schema defines the multiplicity of relationships. Don’t use the instances of the schema to determine multiplicity.
Multiplicity of a Binary Relationship

- \( R \) is a \textit{\textbf{many-one relationship}} from \( E \) to \( F \) if each entity in \( E \) can be connected to \textbf{at most} one entity in \( F \) by \( R \).
  - “at most one” and not “at least one” or “exactly one”.
- \( R \) is \textit{\textbf{one-one}} if it is many-one from \( E \) to \( F \) and many-one from \( F \) to \( E \).
- Otherwise, \( R \) is a \textit{\textbf{many-many}} relationship.
- The schema defines the multiplicity of relationships. Don’t use the instances of the schema to determine multiplicity.