Entity-Relationship Modeling

Anne Denton

Department of Computer Science
North Dakota State University

Outline

Table of Contents

Limitations of the Relational Model

- The relational model lacks expressiveness
- Only two modeling concepts
  - Relations with rows and columns
  - Constraints
- Relations can be entities, relationships, and even some attributes (e.g., multi-valued ones)
  - Relationships may or may not require a separate relation
  - Attributes may or may not require a separate relation
  - Even two separate entities may or may not require separate relations

Four Modeling Steps

1. Requirements collection
   - Maybe the most challenging part!
2. Conceptual modeling
   - Translation of real-world expectations into a data model
   - Course topic: Entity-relationship modeling
3. Data model mapping (logical design)
   - Translation of entity-relationship model to relational model
   - Course topic: Relational modeling
4. Physical modeling
   - Define internal storage structures and access paths
   - Course topic: Indexes

Step 1: Requirements collection

- **Requirements collection is where problems most commonly originate!**
- Make sure you understand all requirements and constraints
- Write down all assumptions you make, and discuss them with the users
- Check later if your design meets their requirements
- Types of information needed:
  - Data requirements (go into conceptual design)
  - Functional requirements (user defined operations and transactions)

Step 2: Conceptual modeling

- Create a conceptual design for the data requirements
  - Conventional modeling language: Entity-relationship modeling
  - Modern alternative: UML class diagrams
- Check your design
  - Are data requirements met?
  - Can functional requirements be met?
  - Confirm that your assumptions were valid!

Step 2 cont.: Alternatives for Conceptual Modeling Languages

- Conventional Entity-Relationship diagrams
  - Entities represented as boxes that are surrounded by bubbles for attributes
  - Relationships rhombus-shaped
- Universal Modeling Language (UML) Class Diagrams
  - Look more familiar and are more practical
  - Entities in databases are mostly like classes in object-oriented programming
  - Main difference between relational and object-oriented design is in the role of data hiding

Question 1 (Multiple answers can be correct)

When you design the conceptual schema of a database (as opposed to at some later time)

1. Make sure that all relevant information is represented, not just the information for a subset of user groups
2. Make sure that the storage media you are assigning will be large enough
3. Make sure not to give financial managers too much access
4. Make sure that you are using appropriate design principles for conceptual design

Step 3: Data Model Mapping (Logical Design)

- Tools exist for automatic mapping from object-oriented design to relational model
  - Computer-Aided Software Engineering (CASE) tools
  - Unfortunately either expensive, or limited in capabilities, or both
- Can also be done dynamically, using object-relational mapping
  - `https://en.wikipedia.org/wiki/Object-relational_mapping`
  - Creates virtual object database
  - Part of many web frameworks, like Django

Step 4: Physical Design

- Define internal storage structures, access paths, etc.
- Examples of access paths
  - Index: Conceptually like index of a book
  - Will be discussed later in the course
- This design may have to be revisited after performance analysis
- Application software / middleware should not depend on the physical design

Table of Contents

Basic Concepts

- **Entity:** A "thing" or "person" in the real world (e.g., an employee)
- **Attribute:** A property of an entity (e.g., an employee's name)
- **Relationship:** A link between two entities (e.g. department with employees)

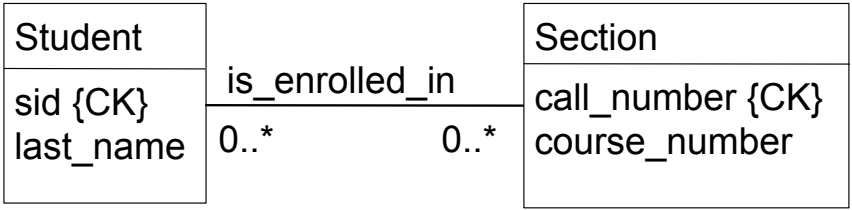
How About Primary Keys?

- What is a primary key?
  - Attribute(s) used to identify a record in a database
  - Must be unique and must not be null
  - Important constraint in a database!
- Should you designate primary keys during conceptual design?
  - Any information on constraints is valuable
  - Use “{CK}" to signify that no duplicates are allowed and the attribute cannot be null, i.e., it is a candidate key
  - That will indicate that a uniqueness constraint and constraint on null should be imposed later
  - “{not null}" and “{unique}" can also be specified separately
- Traditional ER modeling included designating a primary key
  - **But:** Use of real-world attributes as primary key is now discouraged
  - Designating a primary will be part of the data model mapping

UML-style notation

- Based on UML class diagrams
- Entities and attributes
- Entities together with attributes surrounded by a box
  - Use singular of nouns
  - Entities capitalized
  - Attributes lower case
- Relationships
  - Modeled as associations
  - UML also knows aggregations for which one entity is a component part of the other
  - No difference between associations and aggregations in relational mapping, so we use the association notation
  - Most important information: multiplicity
  - Relationships should be named, typically using verbs

Basic example



Comparison with Microsoft Access

- Microsoft Access largely uses ER notation for relational diagrams
- Only has the flexibility of relational modeling!
- For example, many-to-many relationships cannot be captured (also multi-valued attributes, etc.)

Table of Contents

Multi-valued vs. Single-valued attributes

- Single-valued attributes can only have one value
- Multi-valued attributes can have more than one value
- Notation: Multiplicity in brackets
  - Could be a fixed interval [1..3]
  - * used for arbitrarily large numbers [0..*] or [1..*]
- The relational model only knows single-valued attributes!

- Example of a multi-valued attribute: “prior_degrees" that a student completed

Student
sid {CK} last_name prior_degrees [0..*]

- Example of a single-valued: “birth_date"?

Question 2 (Multiple answers can be correct)

The birth date of a person

1. Must be modeled as a single-valued attribute
2. Will normally be modeled as a single-valued attribute
3. As with any piece of information, the modeling may depend on the specific application
4. Will normally be modeled as a multi-valued attribute

Composite vs. Atomic Attributes

- Composite attributes can be subdivided into smaller parts
  - Example: “Name" can be subdivided into “firstName", “middleInitial" and “lastName"
- Indented in diagram
- Allows modeling related attributes together to clarify design
- The relational model only knows atomic attributes!

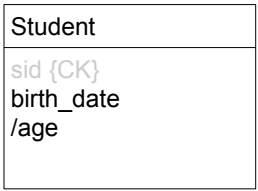
Student
sid {CK} name first_name middle_initial last_name

Complex Attributes

- Composite and multi-valued attributes can be nested
- Complex attributes, like any other can also be modeled as separate entities
  - For clarity, we will treat them as separate entities

Derived vs. Stored Attributes

- Derived attributes can be calculated / deduced from others
- Examples
  - "Age" can be calculated from "BirthDate" and "CurrentDate"
  - "NumberOfStudents" in a course can be evaluated from the student records
- Notation: Slash in front of name (/attributeName)
- Not represented in relational model!
- The relational model only includes stored attributes!



Question 3 (Multiple answers can be correct)

The number of credit hours a student has taken

1. Should not be stored in a database table if it can be derived from enrollment information
2. May be listed in an entity-relationship diagram, even if it can be derived from enrollment information, provided it is identified as a derived attribute
3. Before deciding on whether it is a derived attribute, find out if there ever is a reason to manually override the value

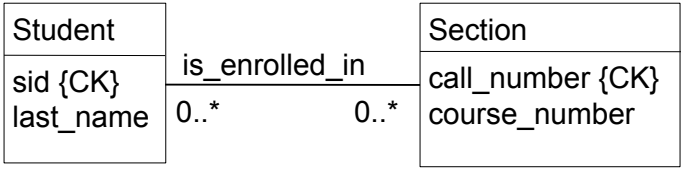
Constraints on Attributes

- Uniqueness
  - Some attributes uniquely identify entity instances
  - For example, no two NDSU students have the same ID
  - Enforcing constraints prevents entering the same entity instance twice
- Not Null
  - A unique attribute that can be null would not identify every attribute
  - Some non-unique attributes are also expected to have a value
- Add {CK} for attributes that are unique and not null
- You can also specify {unique} and {not null} individually

Table of Contents

Relationships

- Form associations of entities
- Examples
  - A `Course` has `Sections`
  - A `Student` enrolls in a `Section`
- Why not use an attribute?
  - `Department` would also have to have `Course` as an attribute
- Most relationships relate two entities, i.e. are binary relationships (others will be discussed later)



Cardinality ratios

- Maximum number of entities involved on each side of a binary relationship
- Modeling constraint that affects the mapping to the relational model
- Can be 1, many (*), or any other value
- Number of entity instances on that side that can be related to any one on the other side

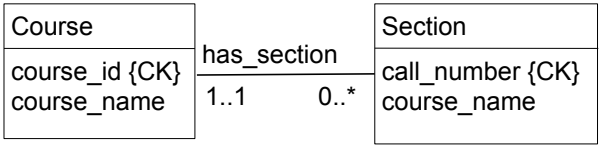
One–one (1:1)

- Each of the two entities is associated with one of the other
- Rightmost of the numbers in the UML diagram 0..**1**, or 1..**1**
  Example: Students having a BisonBucks account

One–many (1:*)

- One of the entities is associated with arbitrarily many of the other
- Again, note the rightmost symbol in the UML diagram 1..**1**, 0..*****

Example: Course having potentially multiple sections



Question 4 (Multiple answers can be correct)

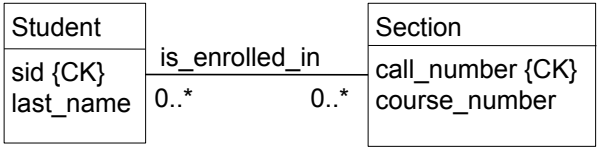
The relationship customer – order should be 1:many

1. If the customer can only order the item once (such as with a club membership)
2. If it is important that the purchase is associated with all owners such as with a house purchase
3. If it is guaranteed that only one customer will be listed in the order

Many–many (*:*)

- Each of the entities may be associated with arbitrarily many of the respective other
- In the UML diagram 0..*, 1..*

Example: One students may be enrolled in multiple sections and each section typically has multiple students enrolled



Participation Constraints

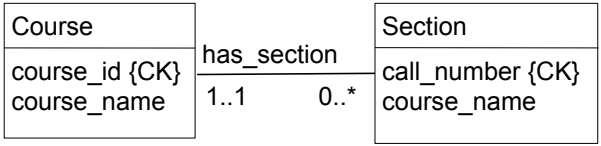
- Defines if an entity has to participate in a relationship
  - Mandatory participation: All entity occurrences involved in relationship
  - Optional participation: Only some entity occurrences involved
- Modeling constraint
  - May affects relational modeling
  - May suggest creating constraints in the relational model
  - Especially important when cardinality is 1, i.e., the instance is mandatory vs. the instance is optional
  - Not as often enforced for higher cardinality, i.e., at least one instance vs. zero or more

Participation Constraint Example

- In the following example a student does not have to have a BisonBucks accounts
- But the BisonBucks account has to belong to a student

Example

- Sections can only exist if they represent a course ⇒ mandatory participation of course
- Courses can exist without sections ⇒ optional participation of section
- Notice that optional participation of



- In other words:
  - The existence of sections depends on the existence of a course
  - The existence of courses does not depend on the existence of a section

Question 5 (Multiple answers can be correct)

Consider an airport database. Should an airplane be required to be of a particular model? (mandatory participation of airplane model)

1. Mandatory participation is correct since it would be dangerous to allow airplanes that are not fit a standard model
2. It is important to determine whether the airplane model information is required even for self-designed airplanes
3. Mandatory participation should be avoided because flexibility is preferable
4. If participation were to be made mandatory, a model would have to be specified for every self-designed airplane
5. If it isn't, self-designed airplanes could be stored without defining a model

Attributes on a relationship

- Notice that relationships can also have attributes:

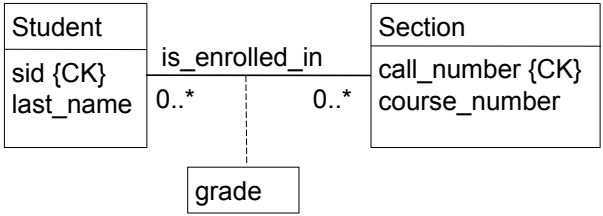


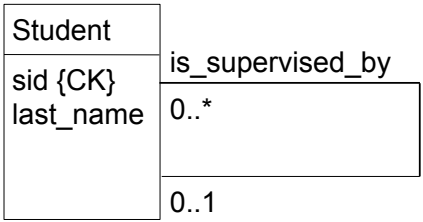
Table of Contents

Weak entity types

- Weak entity types are identified by their relationship to a strong entity type (owner entity type)
- Always have a total participation constraint with respect to their identifying relationship
- Since we do not designate primary keys in the conceptual modeling, there is no difference to other entities that have total participation with regard to one other entity

Recursive relationships

- Relationships can, and often do, connect one entity with itself
  - E.g., an employee may have a supervisor who is also an employee, or web pages hyper-linking to other web pages
- Sometimes called "unary" in contrast to the normal “binary" relationships and higher order ones (next slide)
- We will use the term “Recursive relationship" to avoid confusion with n-ary relations



Ternary relationships

- Relationships that relate three entities are also possible
- When they are used in UML modeling, a rhombus is often used as in conventional ER modeling
- We will model these as a separate entity that represents the relationship
- Example
  - Student `s` majors in Program `p` during Term `t`
- Conventional representation: rhombus-shaped descriptor
- Alternatively you can create work around ternary relationships by creating a new entity, in the example an enrollment entity
- Higher orders also possible

Table of Contents

Example

- A company is organized into departments. Each department has a unique identifier, a name, and is managed by one particular employee. A department may have several locations.
- A department controls a number of projects, each of which has a unique id, a name, and the number of employees involved.
- The database will store each employee's name, social security number, address, salary, and birth date. An employee is assigned to one department, but may work on several projects that may not be controlled by the same department. An employee has one direct supervisor, who is also an employee.
- The database will keep track of dependents, if any, for each employee, for insurance purposes, including each dependent's first name, birth data, and relationship to the employee.

Question 6 (Multiple answers can be correct)

Consider the company database description. The locations of the department

1. Could be modeled as a single-valued attribute
2. Could be modeled as a multi-valued attribute
3. Could be modeled as a derived attribute
4. Could be modeled as a separate entity that is connected to the department via a 1:many relationship

Question 7 (Multiple answers can be correct)

Consider the company database description. The name attribute for the employee entity

1. May be a composite attribute that consists of multiple pieces
2. Is described as a simple attribute, i.e. all parts should be stored in one field
3. Is a derived attribute given the social security number

Question 8 (Multiple answers can be correct)

Consider the company database description. The project – employee relationship

1. Could be modeled as a 1:many relationship if multiple employees work on a project
2. Could be modeled as a many:many relationship if multiple employees work on a project
3. Could be modeled as a 1:1 relationship if an employee works on a project
4. Could be modeled as a many:1 relationship if no project involved more than one employee

Question 9 (Multiple answers can be correct)

Consider the company database description. The managed-by relationship

1. Could be modeled as a many:1 relationship if the same manager could manage multiple departments (to answer this question consider department membership of employees)
2. Could be modeled as a 1:1 relationship if no manager is assigned to more than one department and the manager is part of the department he or she manages
3. Could be modeled as having total employee participation, i.e. all departments must have a manager
4. Could be modeled as having total department participation, i.e. all employees must manage a department

Question 10 (Multiple answers can be correct)

Consider company database description. The has-dependent relationship

1. Could be modeled as a 1:many relationship if the assumption is made that a dependent is connected with exactly one employee
2. Could be modeled as a many:many relationship if dependents can potentially be connected with multiple employees
3. Could be modeled as having total dependent participation, i.e. all employees must be connected with at least one dependent
4. Could be modeled as having total employee participation, i.e. all dependents must be connected with an employee

Example Diagram