Conceptual Design with ER Model

Lecture #2
1/24/2012

Jeff Ballard
CS564, Spring 2014, Database Management Systems
Homework 1 is out

• See the Moodle page
• Due February 7
• Groups of 2-3 people
  – Pick a team name
  – Individual groups are acceptable
    • If you choose to work alone, be warned, various phases of this project are a lot of work and you will be expected to complete all tasks
Lecture Outline

- Steps in building a database application
- Conceptual design with ER model
Steps in Building a DB Application

• Step 0: pick an application domain
• Step 1: conceptual design
  – discuss what to model in the application domain
  – need a modeling language to express what you want
  – ER model is the most popular such language
  – output: an ER diagram of the application domain
Steps in Building a DB Application

• Step 2: pick a type of DBMS
  – relational DBMS is most popular and is our focus

• Step 3: translate ER design to a relational schema
  – use a set of rules to translate from ER to rel. schema
  – use a set of schema refinement rules to transform the above rel. schema into a good rel. schema

• At this point
  – you have a good relational schema on paper
Steps in Building a DB Application

• Subsequent steps include
  – implement your relational DBMS using a "database programming language" called SQL
  – ordinary users cannot interact with the database directly
  – and the database also cannot do everything you want
  – hence write your application program in C++, Java, Perl, etc to handle the interaction and take care of things that the database cannot do

• So, the first thing we should start with is to learn ER model ...
ER Model

• Gives us a language to specify
  – what information the database must hold
  – what are the relationships among components of that information

• Proposed by Peter Chen in 1976

• What we will cover
  – basic stuff
  – constraints
  – weak entity sets
  – design principles
Basic Concepts
Entities and Attributes

• Entities
  – real-world objects distinguishable from other objects
  – described using a set of attributes

  - price
  - name
  - category

  Product

  - name
  - stockprice

  Company

• Attributes
  – each has an atomic domain: string, integers, reals, etc.

• Entity set: a collection of similar entities
Relations

• A mathematical definition:
  – if A, B are sets, then a relation R is a subset of $A \times B$

• $A=\{1,2,3\}$, $B=\{a,b,c,d\}$,
  $R = \{(1,a), (1,c), (3,b)\}$

makes is a subset of $\text{Product} \times \text{Company}$:
More about relationships ...
Multiplicity of E/R Relations

• one-one:

• many-one

• many-many
Important: Difference with the Book

- We will use

  - Cow book uses (see Page 33, 3rd edition)

- You should use the notation given in class, not the book!
Multiway Relationships

How do we model a purchase relationship between buyers, products and stores?

Can still model as a mathematical set (how?)
Arrows in Multiway Relationships

Q: what does the arrow mean?

A: if I know the store, person, invoice, I know the movie too
Q: what do these arrows mean?

A: store, person, invoice determines movie and store, invoice, movie determines person
Q: how do I say: “invoice determines store”?
A: no good way; best approximation:

Q: Why is this incomplete?
Roles in Relationships

What if we need an entity set twice in one relationship?

Product

Purchase

Store

Person

salesperson

buyer

Person
Roles in Relationships

What if we need an entity set twice in one relationship?

- Product
- Person
- Store

Roles:
- Purchase
  - salesperson
  - buyer
Attributes on Relationships
Converting Multiway Relationships to Binary

- Purchase
  - Date
  - StoreOf
    - Store
  - BuyerOf
    - Person
  - ProductOf
    - Product
Relationships: Summary

- Modeled as a mathematical set
- Binary and multiway relationships
- Converting a multiway one into many binary ones
- Constraints on the degree of the relationship
  - many-one, one-one, many-many
  - limitations of arrows
- Attributes of relationships
  - not necessary, but useful
Subclasses

• Subclass = special case = fewer entities = more properties.

• Example: Ales are a kind of beer.
  – Not every beer is an ale, but some are.
  – Let us suppose that in addition to all the properties (attributes and relationships) of beers, ales also have the attribute color.
Subclasses in ER Diagrams

• Assume subclasses form a tree.
  – I.e., no multiple inheritance.

• Isa triangles indicate the subclass relationship.
  – Point to the superclass.
Example

![Entity-Relationship Diagram]

- **Beers**
  - **name**
  - **manf**
  - **isa**
  - **color**

- **Ales**

The diagram represents a subset of characteristics and relationships within the broader concept of Beers.
ER Vs. Object Oriented Subclasses

• In the object-oriented world, objects are in one class only.
  – Subclasses inherit properties from superclasses.
• In contrast, E/R entities have components in all subclasses to which they belong.
  – Matters when we convert to relations.
Example

Beers

Pete’s Ale

name

manf

isa

color

Ales
Constraints

• A constraint = an assertion about the database that must be true at all times
• Part of the database schema
• Very important in database design
Modeling Constraints

Finding constraints is part of the modeling process. Commonly used constraints:

**Keys:** Student ID number uniquely identifies a person.

**Single-value constraints:** a person can have only one biological father.

**Referential integrity constraints:** if you work for a company, it must exist in the database.

**Domain constraints:** peoples’ ages are between 0 and 150.

**General constraints:** all others (at most 75 students enroll in a class)
Why Constraints are Important?

• Give more semantics to the data
  – help us better understand it

• Allow us to refer to entities (e.g, using keys)

• Enable efficient storage, data lookup, etc.
Keys in E/R Diagrams

Underline:

No formal way to specify multiple keys in E/R diagrams
More about Keys

• Every entity set must have a key
  – why?

• A key can consist of more than one attribute

• There can be more than one key for an entity set
  – one key will be designated as primary key

• Requirement for key in an isa hierarchy
  – not covered in this lecture
Important: Difference with the Book

• Recall that when we have:

  ![Diagram: Employee Manages Department](image)

• Cow book will write (see Page 33, 3rd edition):

  ![Diagram: Employee Manages Department](image)

• And interprets this as a key constraint
Important: Difference with the Book

• Why?

• Because given a department ID, we can uniquely identify the manage relationship it participates in:
  - (emp1, dept1)
  - (emp1, dept2)
  - (emp2, dept3)
  - (emp3, dept1)


Important: Difference with the Book

• However, we view this as many-one constraint

• For us, key constraint = set of attributes that uniquely identify entities in an entity set
Single Value Constraint

• At most one value plays a particular role

• An attribute of a entity set has a single value
  – we can specify if the value must be present or can be missing (represented with say NULL or -1)
  – example in real-estate domain
    • price vs. house-style

• A many-one relation implies single value constraint
Referential Integrity Constraint

• Single value constraint: at most one value exists in a given role

• Referential integrity constraint: exactly one value exists in a given role

• An attribute has a non-NULL, single value
  – this can be considered a kind of referential integrity constraint

• However, we more commonly use such constraints to refer to relationships
Referential Integrity Constraints

• In some formalisms we may refer to another object but get garbage instead
  – e.g. a dangling pointer in C/C++

• the Referential Integrity Constraint on relationships explicitly requires a reference to exist
• This will be even clearer once we get to relational databases
Other Kinds of Constraints

- Domain constraints
- Constraints on degree of a relationship
- Other more general constraints
- See the book
Weak Entity Sets

Entity sets are weak when their key attributes come from other classes to which they are related.

This happens if:

- part-of hierarchies
- splitting n-ary relations to binary.
Weak Entity Sets

• Occasionally, entities of an entity set need “help” to identify them uniquely.

• Entity set $E$ is said to be *weak* if in order to identify entities of $E$ uniquely, we need to follow one or more many-one relationships from $E$ and include the key of the related entities from the connected entity sets.
Now, about design techniques ...
Design Principles 1: Be Faithful
Design Principles 2: Avoid Redundancy

- Product
- Purchase
- date
- personName
- personAddr
- Store
Design Principles 3: KISS

- Read the book for more design principles
More on Design Techniques
(from Ullman's slides)

1. Avoid redundancy.
2. Limit the use of weak entity sets.
3. Don’t use an entity set when an attribute will do.
Avoiding Redundancy

• Redundancy occurs when we say the same thing in two different ways.
• Redundancy wastes space and (more importantly) encourages inconsistency.
  – The two instances of the same fact may become inconsistent if we change one and forget to change the other, related version.
Example: Good

This design gives the address of each manufacturer exactly once.
Example: Bad

This design states the manufacturer of a beer twice: as an attribute and as a related entity.
Example: Bad

This design repeats the manufacturer’s address once for each beer; loses the address if there are temporarily no beers for a manufacturer.
Entity Sets Versus Attributes

• An entity set should satisfy at least one of the following conditions:
  – It is more than the name of something; it has at least one non-key attribute.
    or
  – It is the “many” in a many-one or many-many relationship.
- *Manfs* deserves to be an entity set because of the nonkey attribute *addr*.

- *Beers* deserves to be an entity set because it is the "many" of the many-one relationship *ManfBy*. 
Example: Good

There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.
Example: Bad

Since the manufacturer is nothing but a name, and is not at the “many” end of any relationship, it should not be an entity set.
Don’t Overuse Weak Entity Sets

• Beginning database designers often doubt that anything could be a key by itself.
  – They make all entity sets weak, supported by all other entity sets to which they are linked.

• In reality, we usually create unique ID’s for entity sets.
  – Examples include social-security numbers, student-ID numbers, automobile VIN’s, etc.
When Do We Need Weak Entity Sets?

• The usual reason is that there is no global authority capable of creating unique ID’s.
• Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.
ER Review

• Basic stuff
  – entity, attribute, entity set
  – relation: binary, multiway, converting from multiway
  – relationship roles, attributes on relationships
  – subclasses (is-a)

• Constraints
  – on relations
    • many-one, one-one, many-many
    • limitations of arrows
  – keys, single-valued, ref integrity, domain & general constraints
ER Review

- Weak entity sets
- Design principles
  - be faithful
  - avoid redundancy
  - KISS