Introduction to Database Systems
CSE 444
Lecture #7
Jan 24 2001

Announcements
* Programming Assignment due tomorrow [Thu (1/25)]
* Mid Term Syllabus
  - Material in lectures
  - Textbook
    - Chapter 1, Chapter 2 (except 2.1 and ODL)
    - Chapter 3 (except 3.2), Chapter 4 (except 4.2, 4.3)
    - Chapter 5 (except 5.10)
    - Chapter 6, Chapter 7 (except 7.2)
  - Mid Term will be in class closed book exam

Database Schema Design
Today's Reading:
Sec 2 (except 2.1 and ODL discussions) and
Sec 3.1-3.4 (except 3.1)

Overview of Database Schema Design
* Conceptual design (ER Model)
  - ER Diagram
    - What are the entities and relationships in the enterprise?
    - What are the integrity constraints or business rules that hold?
    - Map an ER diagram into a relational schema
* Schema Refinement (Normalization):
  - Check relational schema for redundancies and related anomalies
* Physical Design:
  - Determine physical structures

ER Model Basics
* Entity: Real-world object distinguishable from other objects. An entity is described (in DB) using a set of attributes.
* Entity Set: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes.
  - Each entity set has a key.
  - Each attribute has a domain.

ER Model Basics
* Entity and Entity Set
  - Attributes (atomic but may be null)
* Relationship and Relationship Set
  - Attributes (atomic)
### Multiplicity of E/R Relationships

- **one-one:**
- **many-one:**
- **many-many:**

### Multi-way Relationship

#### Arrows in Multiway Relationships (1)

**Q:** what does the arrow mean?

**A:** store, person, invoice determines the movie

#### Arrows in Multiway Relationships (2)

**Q:** what do these arrows mean?

**A:** store, person, invoice determines movie and store, invoice, movie determines person

#### Arrows in Multiway Relationships (3)

**Q:** how do I say: "invoice determines store"?

**A:** no good way; best approximation:

**Why is this incomplete?**

### Design of ER Models

- Picking the right kind of element requires care
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, roles, ...
  - Read Example 2.17 for illustration
- Some design principles that help:
  - Faithfulness
  - Avoidance of redundancy
  - Simplicity
Example 1: Design Principles

What's wrong?

- Product → Purchase → Person
- Country → President → Person

Moral: be faithful!

Example 2: Design Principles

Product → date
Purchase → Store
personAddr → person

Moral: pick the right kind of elements.

Example 3: Design Principles

Product → Purchase → Store
Dates → date

Moral: don't complicate life more than it already is.

Understanding Subclasses

Think in terms of records:
- Product: field1, field2
- SoftwareProduct: field1, field2, field3
- EducationalProduct: field1, field2, field3

Modeling Subclasses

Some objects in a class may be special:
- define a new class
- better: define a subclass

Products
- Software products
- Educational products

So --- we define subclasses (in ODL and in E/R).
Modeling Union Types With Subclasses

Say: each piece of furniture is owned either by a person, or by a company.

Solution 1. Acceptable, imperfect (What’s wrong?)

Modeling Constraints

Finding constraints is part of the modeling process.

Commonly used constraints:

- Keys: social security number uniquely identifies a person.
- Single-value constraints: a person can have only one 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 50 students enroll in a class)

Keys

A set of attributes that uniquely identify an object or entity:

- Person: social security number, name, address, age
- Product: name, category, price

Perfect keys are often hard to find, so organizations usually invent something.

An object may have multiple keys:

- employee number, social-security number
- address, name, ssn
Single Value Constraints

- Each attribute can only have atomic (single) value
- Many to one relationship

Referential Integrity Constraints

- The Referential Integrity Constraint explicitly requires a reference to exist
  - Specialization of single value constraint
  - Avoids situations where we refer to an object but get garbage instead
    - E.g., a dangling pointer in C/C++

Example: Referential Integrity

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

Modeling Constraints in ER

- Constraints play an important role in determining the best database design for an enterprise
- Several kinds of integrity constraints can be expressed in the ER model:
  - Keys
  - Referential constraints
- Some constraints cannot be expressed in the ER model:
  - Some functional dependencies
  - Domain constraints
Translating E/R Diagrams into Relational Schemas

Reading: Chapters 3.1, 3.3, 3.4

Entity Sets to Relations

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>gizmo</td>
<td>gadgets</td>
<td>$19.99</td>
</tr>
</tbody>
</table>

Relationships to Relations

<table>
<thead>
<tr>
<th>Product-name</th>
<th>Product-Category, Company-name</th>
<th>Starting-year</th>
</tr>
</thead>
<tbody>
<tr>
<td>gizmo</td>
<td>gadgets gizmoWorks</td>
<td>1963</td>
</tr>
</tbody>
</table>

Rules for Binary Relationships

Several cases are possible for a binary relationship:

1. Many-many:
2. Many-one:
3. One-one:
Example: Multiway Relationships

How will you map Purchase to a relation?

Handling Weak Entity Sets

Relation Team:

<table>
<thead>
<tr>
<th>Sport</th>
<th>Number</th>
<th>Affiliated University Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>mud wrestling</td>
<td>15</td>
<td>Montezuma State U.</td>
</tr>
</tbody>
</table>

- need all the attributes that contribute to the key of Team
- don’t need a separate relation for Affiliation (why?)

Modeling Subclass Structure

Mapping Subclasses

Product(name, price, category, manufacturer)
EducationalProduct(name, ageGroup, topic)
SoftwareProduct(name, platforms, requiredMemory)

No need for a relation EducationalSoftwareProduct

Unless, it has a specialized attribute:

EducationalSoftwareProduct(name, educational-method)

Subclasses in E/R Diagrams

SQL for Keys and Reference Keys

CREATE TABLE Books (isbn CHAR(11), title CHAR(20), pubname CHAR(25), pubdate DATE, PRIMARY KEY (isbn), FOREIGN KEY (pubname) REFERENCES Publishers (name))
**E/R to Relations: Summary**

- Entity set → Relation
- M-N Relationship → Relation (keys of related entities plus relationship attributes)
- Special Cases:
  - M-1 Relationship
  - 1-1 Relationship
- The resultant relational schema may have some nasty properties....

**Functional Dependencies**

Reading: Chapter 3.5, 3.6

---

**Motivation**

- Subjective nature of E-R diagram may not capture all relationships
- ER → Relational translation may not satisfactory:
  - Some relationships not easily captured by translation
- Recognizing Functional Dependencies helps refine schema

**Example**

(Employees)

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E1847</td>
<td>John</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>lawyer</td>
</tr>
</tbody>
</table>

**Example (Contd)**

- The specific ER Diagram did not capture that position has a unique telephone number
- What if:
  - All current salespersons resign
  - Can I update Smith’s phone?
  - Can I add a salesperson Roy with phone 6923?

---

**The Evils of Redundancy**

- Redundancy is at the root of these problems:
  - Redundant storage, insert/delete/update anomalies
- Integrity constraints, in particular functional dependencies, can be used to identify problem schema
- Use decomposition judiciously to overcome these issues
  - Replacing ABCD with, say, AB and BCD, or ACD and ABD
**Functional Dependencies**

Definition:

If two tuples agree on the attributes

\[ A_A, A_2, \ldots, A_n \]

then they must also agree on the attributes

\[ B_B, B_2, \ldots, B_m \]

Formally:

\[ A_A, A_2, \ldots, A_n \rightarrow B_B, B_2, \ldots, B_m \]

Motivating example for the study of functional dependencies:

<table>
<thead>
<tr>
<th>Name</th>
<th>Social Security Number</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>John</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>Mary</td>
<td>1234</td>
<td>lawyer</td>
</tr>
</tbody>
</table>

---

**In General**

To check \( A \rightarrow B \), erase all other columns

\[
\begin{array}{c|c}
\ldots & \ldots \\
A & B \\
--- & --- \\
X_1 & Y_1 \\
X_2 & Y_2 \\
\ldots & \ldots \\
\end{array}
\]

check if the remaining relation is many-one (called *functional* in mathematics)

---

**Example**

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E1847</td>
<td>John</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>lawyer</td>
</tr>
</tbody>
</table>

---

**Keys and SuperKeys**

- **Product**: \( \text{name} \rightarrow \text{price, manufacturer} \)
- **Person**: \( \text{ssn} \rightarrow \text{name, age} \)
- **Company**: \( \text{name} \rightarrow \text{stock price, president} \)

Key of a relation is a set of attributes that:
- functionally determines all the attributes of the relation
- none of its subsets determines all the attributes.

Superkey: a set of attributes that contains a key.

---

**A Property of Functional Dependency**

Splitting/Combining Lemma

\[ A_A, A_2, \ldots, A_n \rightarrow B_B, B_2, \ldots, B_m \]

Is equivalent to

\[ A_A, A_2, \ldots, A_n \rightarrow B_1 \\
A_A, A_2, \ldots, A_n \rightarrow B_2 \\
\vdots \\
A_A, A_2, \ldots, A_n \rightarrow B_m \]

---

**Inferring Implied Functional Dependency**

**Reflexivity**

\[ A_A, A_2, \ldots, A_n \rightarrow A_i \]

Always holds

**Why?**
Inferring Implied Functional Dependency (contd.)

**Augmentation Rule:**

If \( A_1, A_2, \ldots, A_n \rightarrow C_1, C_2, \ldots, C_p \)
then \( A_1, A_2, \ldots, A_n \rightarrow B_1 \ldots B K \)
\( C_1, C_2, \ldots, C_p \rightarrow B_1 \ldots B K \)

**Why?**

---

Inferring Implied Functional Dependency (contd.)

**Transitive Closure Rule:**

If \( A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m \)
and \( B_1, B_2, \ldots, B_m \rightarrow C_1, C_2, \ldots, C_p \)
then \( A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m \rightarrow C_1, C_2, \ldots, C_p \)

**Why?**

---

Inference of Implied FD (contd.)

Armstrong’s axioms

- Reflexivity
- Augmentation
- Transitivity

A sound and complete inference rule to obtain all implied functional dependencies

---

Closure of a set of Attributes

Given a set of attributes \( \{A_1, \ldots, A_n\} \) and a set of dependencies \( S \).

Problem: find all attributes \( B \) such that:

any relation which satisfies \( S \) also satisfies:

\( A_1, \ldots, A_n \rightarrow B \)

The closure of \( \{A_1, \ldots, A_n\} \), denoted \( \{A_1, \ldots, A_n\}^+ \),
is the set of all such attributes \( B \)

What is the relationship between closure and keys?

---

Closure Algorithm

Start with \( X = \{A_1, \ldots, A_n\} \).

Repeat until \( X \) doesn’t change do:

if \( B_1, B_2, \ldots, B_n \rightarrow C \) is in \( S \) and
\( B_1, B_2, \ldots, B_n \) are all in \( X \), and
\( C \) is not in \( X \)
then
add \( C \) to \( X \).

---

Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>B</td>
</tr>
</tbody>
</table>

Closure of \{A, B\}: \( X = \{A, B\} \)
Closure of \{A, F\}: \( X = \{A, F\} \)