Lecture 3:
Conceptual Database Design and Schema Design
April 12th, 2004

Agenda
• Project: questions?
• How to model data (E/R modeling)
• How to design a good schema (normalization).

Building an Application with a DBMS
• Requirements modeling (conceptual, pictures)
  - Decide what entities should be part of the application and how they should be linked.
• Schema design and implementation
  - Decide on a set of tables, attributes.
  - Define the tables in the database system.
  - Populate database (insert tuples).
• Write application programs using the DBMS
  - Way easier now that the data management is taken care of.

Database Design
• Why do we need it?
  - Agree on structure of the database before deciding on a particular implementation.
• Consider issues such as:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good designs

Database Design Formalisms
1. Object Definition Language (ODL):
  - Closer in spirit to object-oriented models.
  - I don’t teach it anymore.
2. Entity/Relationship model (E/R):
  - More relational in nature.
• Both can be translated (semi-automatically) to relational schema.
• ODL to OO-schema: direct transformation (C++ or Smalltalk based system).

2. Entity/Relationship Diagrams

Database Design

Entities
Product

Attributes
address

Relationships between entities
buys
Keys in E/R Diagrams

- Every entity set must have a key

```
Product
- name
- category
- price
```


What is a Relation?

- A mathematical definition:
  - If A, B are sets, then a relation R is a subset of A x B
  - \( A = \{1, 2, 3\}, B = \{a, b, c, d\}, R = \{(1, a), (1, c), (3, b)\} \)
  - makes is a subset of Product x Company:

```
Product
- name
- category
- stockprice

Company
- name
- makes
- employs

makes
```

Multiplicity of E/R Relations

- one-one
- many-one
- many-many

Multi-way Relationships

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

```
Purhase
```

Can still model as a mathematical set; how?
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.

Q: why is this incomplete?
A: no good way; best approximation:

Q: how do I say: "invoice determines store"?
A: no good way.

A: if I know the store, person, invoice, I know the movie too.

A: store, person, invoice determines movie.

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

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: why is this incomplete?
A: no good way; best approximation:

Q: how do I say: "invoice determines store"?
A: no good way.

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

A: if I know the store, person, invoice, I know the movie too.

A: store, person, invoice determines movie.

A: if I know the store, person, invoice, I know the movie too.

A: store, person, invoice determines movie.

A: if I know the store, person, invoice, I know the movie too.

A: store, person, invoice determines movie.

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

A: if I know the store, person, invoice, I know the movie too.

A: store, person, invoice determines movie.

A: if I know the store, person, invoice, I know the movie too.

A: store, person, invoice determines movie.

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

A: if I know the store, person, invoice, I know the movie too.

A: store, person, invoice determines movie.

What if we need an entity set twice in one relationship?
From E/R Diagrams to Relational Schema

- Entity set relation
- Relationship relation

Entity Set to Relation

<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

Makes: (product-name, product-category, company-name, year)

Product-name    Product-category  Company-name    Starting-year

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>price</th>
<th>Start Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>gizmo</td>
<td>gadgets</td>
<td>19.99</td>
<td>1963</td>
</tr>
</tbody>
</table>

Multi-way Relationships to Relations

What’s wrong?

Moral: be faithful!
Design Principles: What’s Wrong?

- Moral: pick the right kind of entities.

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

Modeling Subclasses

The world is inherently hierarchical. Some entities are special cases of others.
- We need a notion of subclass.
- This is supported naturally in object-oriented formalism.

Products

Software products

Educational products

Understanding Subclasses

- Think in terms of records:
  - Product
  - SoftwareProduct
  - EducationalProduct
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?)

Solution 2: Better, more laborious

Constraints in E/R Diagrams

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.

Other constraints: people’s ages are between 0 and 150.

Keys in E/R Diagrams

No formal way to specify multiple keys in E/R diagrams.

Single Value Constraints
Referential Integrity Constraints

Other Constraints

Weak Entity Sets
Entity sets are weak when their key comes from other classes to which they are related.

Handling Weak Entity Sets
Convert to a relational schema (in class)

The Relational Data Model

Relational Schema Design
Conceptual Model:
Relational Model:
Normalization:
Eliminates non-keys

Physical storage

E/R diagrams
Tables: columns: attributes row: tuples

Complex file organization and index structures.

Data Modeling
Relational Schema
Physical storage

Person buys Product
name price

name

**Functional Dependencies**

Definition: \( A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n \) holds in \( R \) if:

\[
\forall t, t', \in R, (t.A_1 = t'.A_1 \land \ldots \land t.A_m = t'.A_m) \Rightarrow (t.B_1 = t'.B_1 \land \ldots \land t.B_n = t'.B_n)
\]

**Important Point!**

- Functional dependencies are part of the schema!
- They constrain the possible legal data instances.
- At any point in time, the actual database may satisfy additional FD’s.

**Examples**

- **EmpID** | **Name** | **Phone** | **Position**
  - E0045 | Smith | 1234 | Clerk
  - E1847 | John | 9876 | Salesrep
  - E1111 | Smith | 9876 | Salesrep
  - E9999 | Mary | 1234 | Lawyer

- **Examples of Keys**
  - Product(name, price, category, color)
  - name, category, price
  - category, color
  - Keys: (name, category) and all supersets
  - Enrollment(student, address, course, room, time)
  - student, address, course, room, time
  - Keys: [in class]

- **Finding the Keys of a Relation**

Given a relation constructed from an E/R diagram, what is its key?

**Rules:**

1. If the relation comes from an entity set, the key of the relation is the set of attributes which is the key of the entity set.
Finding the Keys

Rules:
2. If the relation comes from a many-to-many relationship, the key of the relation is the set of all attribute keys in the relations corresponding to the entity sets.

Expressing Dependencies

Say: “the CreditCard determines the Person”

Relational Schema Design (or Logical Design)

Main idea:
- Start with some relational schema
- Find out its FD’s
- It’s important also to look at inferred FD’s.
- Use them to design a better relational schema

Inference Rules for FD’s

\[ A_1, A_2, \ldots, A_m \rightarrow B_1, B_2, \ldots, B_n \]

Is equivalent to

\[ A_1, A_2, \ldots, A_m, A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_n \]

Splitting rule

Combining rule
Inference Rules for FD’s (continued)

\[ A_i \rightarrow A_i \]

Trivial Rule

Where \( i = 1, 2, \ldots, n \)

Why?

---

Transitive Closure Rule

\[ A_i \rightarrow A_i \quad \text{and} \quad B_j \rightarrow B_j \quad \text{then} \quad C_j \rightarrow C_j \]

Why?

---

Enrollment (student, major, course, room, time)

Student major

Major, course room

Course time

What else can we infer? [in class]

---

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 \( \overline{\{A_1, \ldots, A_n\}} \),

is the set of all such attributes \( B \)

The closure tells us everything we can infer from \( A_1, \ldots, A_n \).

---

Closure Algorithm

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

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

- If \( B_i, B_j \rightarrow B_k \) \( \in S \) and \( B_i, B_j \rightarrow B_k \) are all in \( X \), and \( B_k \) is not in \( X \)

then

add \( C \) to \( X \).
Example

\[
\begin{array}{ccc}
A & B & C \\
A & D & E \\
B & D & \\
A & F & B \\
\end{array}
\]

Closure of \{A, B\}: \ X = \{A, B, \}

Closure of \{A, F\}: \ X = \{A, F, \}

Why Is the Algorithm Correct?

- Show the following by induction:
  - \( A_1, \ldots, A_n \) \in \( X \)
  - Initially \( X = \{A_1, \ldots, A_n\} \) \holds
  - Induction step: \( B_1, \ldots, B_m \) \in \( X \)
    - We also have \( B_1, \ldots, B_m \) \in \( C \)
    - By transitivity we have \( A_1, \ldots, A_n \) \in \( C \)
  - This shows that the algorithm is sound; need to show it is complete

Relational Schema Design (or Logical Design)

Main idea:
- Start with some relational schema
- Find out its FD's
- Use them to design a better relational schema

Anomalies:
- Redundancy = repeat data
- Update anomalies = Fred moves to "Bellvue"
- Deletion anomalies = Fred drops all phone numbers: what is his city?

Relation Decomposition

Break the relation into two:

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>908-555-1234</td>
<td>Joe</td>
<td>Westfield</td>
</tr>
<tr>
<td>908-555-2121</td>
<td>Joe</td>
<td>Westfield</td>
</tr>
<tr>
<td>206-555-6543</td>
<td>Fred</td>
<td>Seattle</td>
</tr>
<tr>
<td>206-555-1234</td>
<td>Fred</td>
<td>Seattle</td>
</tr>
</tbody>
</table>

Relation Decomposition

4. Conceptual Model:

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>908-555-1234</td>
<td>Westfield</td>
</tr>
<tr>
<td>Joe</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
<tr>
<td>Fred</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
</tbody>
</table>

5. Relational Model:

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>908-555-1234</td>
<td>Joe</td>
<td>Westfield</td>
</tr>
<tr>
<td>908-555-2121</td>
<td>Joe</td>
<td>Westfield</td>
</tr>
<tr>
<td>206-555-6543</td>
<td>Fred</td>
<td>Seattle</td>
</tr>
<tr>
<td>206-555-1234</td>
<td>Fred</td>
<td>Seattle</td>
</tr>
</tbody>
</table>

6. Normalization:

- Removes anomalies

### Relational Schema Design

<table>
<thead>
<tr>
<th>Conceptual Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Joe</td>
</tr>
<tr>
<td>Joe</td>
</tr>
<tr>
<td>Fred</td>
</tr>
<tr>
<td>Fred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relational Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>908-555-1234</td>
</tr>
<tr>
<td>908-555-2121</td>
</tr>
<tr>
<td>206-555-6543</td>
</tr>
<tr>
<td>206-555-1234</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removes anomalies</td>
</tr>
</tbody>
</table>
Decom positions in General

Create two relations $R_1(B_1, ..., B_m)$ and $R_2(C_1, ..., C_p)$ such that:

$B_1, ..., B_m \subseteq C_1, ..., C_p = A_1, ..., A_n$

and:

$R_1 = \text{projection of } R \text{ on } B_1, ..., B_m$

$R_2 = \text{projection of } R \text{ on } C_1, ..., C_p$

Incorrect Decom position

- Sometimes it is incorrect:

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>DoubleClick</td>
<td>29.99</td>
</tr>
<tr>
<td>Camera</td>
<td>OneClick</td>
<td>24.99</td>
</tr>
<tr>
<td>Gadget</td>
<td>Gizmo</td>
<td>19.99</td>
</tr>
</tbody>
</table>
```

Incorrect Decom position

Decom pose on: Name, Category, and Price.

- Cannot recover information

Normal Forms

- First Normal Form (1NF) = all attributes are atomic
- Second Normal Form (2NF) = old and obsolete
- Third Normal Form (3NF) = this lecture
- Boyce-Codd Normal Form (BCNF) = this lecture
- Others...

Boyce-Codd Normal Form

- A simple condition for removing anomalies from relations:

  A relation $R$ is in BCNF if:

  Whenever there is a nontrivial dependency $A_1, ..., A_n \rightarrow B$ in $R$, $\{A_1, ..., A_n\}$ is a key for $R$.

  In English (though a bit vague):

  Whenever a set of attributes of $R$ is determining another attribute, should determine all the attributes of $R$.

Example

```
<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>HomePhone</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>908-555-1234</td>
<td>987-654-3210</td>
<td>Seattle</td>
</tr>
<tr>
<td>Ted</td>
<td>212-45-6789</td>
<td>212-456-7890</td>
<td>Boston</td>
</tr>
<tr>
<td>Joe</td>
<td>907-456-7890</td>
<td>907-654-3210</td>
<td>Seattle</td>
</tr>
</tbody>
</table>
```

- What are the dependencies?
  - SSN Name, City
  - What are the keys?
    - $\{\text{SSN}, \text{Phone}\}$ unbes
    - Is it in BCNF?
Decompose it into BCNF

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>122-43-7890</td>
<td>Joe</td>
<td>Seattle</td>
</tr>
<tr>
<td>123-45-6789</td>
<td>Fred</td>
<td>City</td>
</tr>
</tbody>
</table>

Summary of BCNF

Decomposition

Find a dependency that violates the BCNF condition:

\[ A_1, A_2, ..., A_n \rightarrow B_1, B_2, ..., B_m \]

Heuristics: choose \( B_1, B_2, ..., B_m \) "as large as possible"

Decom pose:

Continue until there are no BCNF violations left.

Example Decomposition

Person (name, SSN, age, hairColor, phoneNumber)

Decompose in BCNF (in class):

Step 1: find all keys

Step 2: now decompose

Other Example

\[ R(A, B, C, D) \rightarrow A \rightarrow B, B \rightarrow C \]

Key: \( A, D \)

Violations of BCNF: \( A \rightarrow B, A \rightarrow C, A \rightarrow BC \)

Pick \( A \rightarrow BC \): split into \( R1(A, BC) \), \( R2(A, D) \)

What happens if we pick \( A \rightarrow B \) first?

Correct Decompositions

A decom position is lossless if we can recover:

Given \( R \subseteq A \times B \), the decom position into \( R1(A, B), R2(A, C) \) is lossless.

Correct Decompositions

\[ R \in \text{general, larger than } R. \text{ Must ensure } R' = R. \]
### 3NF: A Problem with BCNF

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unit</td>
<td>Company</td>
</tr>
</tbody>
</table>

FD’s: Unit → Company, Company, Product → Unit

So, there is a BCNF violation, and we decompose.

### So What’s the Problem?

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galaga99</td>
<td>UW</td>
<td>databases</td>
</tr>
<tr>
<td>Bingo</td>
<td>UW</td>
<td>databases</td>
</tr>
</tbody>
</table>

No problem so far. All local FD’s are satisfied.

Let’s put all the data back into a single table again:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galaga99</td>
<td>UW</td>
<td>databases</td>
</tr>
<tr>
<td>Bingo</td>
<td>UW</td>
<td>databases</td>
</tr>
</tbody>
</table>

Violates the dependency: Company, Product → Unit!

### Solution: 3rd Normal Form (3NF)

A simple condition for removing anomalies from relations:

A relation R is in 3nd normal form if:

- Whenever there is a nontrivial dependency A₁, A₂, ..., Aₙ → B for R, then {A₁, A₂, ..., Aₙ} is a super-key for R, or B is part of a key.