Today

- Wrap up BCNF
- 3NF
- Intro to ER modeling

Assignments

- HW8 due
- HW9 out
- Proj 2 out
- Exam 2 now Thurs, Nov 15th
Dependency-Preserving Decompositions

Decompositions should also **preserve dependencies**

- For example:

  \[
  \text{Employee}(\text{eid}, \text{address}, \text{city}, \text{state}, \text{zip})
  \]

  \[
  \begin{align*}
  &- \text{zip} \rightarrow \text{state} \quad \text{(note not totally accurate)} \\
  &- \text{address, city, state} \rightarrow \text{zip}
  \end{align*}
  \]

- A possible decomposition

  \[
  \begin{align*}
  \text{Employee}(\text{eid}, \text{address}, \text{city}, \text{zip}) \\
  \text{ZipState}(\text{zip}, \text{state})
  \end{align*}
  \]

- While in BCNF, it does not preserve the FD:

  \[
  \begin{align*}
  &- \text{address, city, state} \rightarrow \text{zip}
  \end{align*}
  \]

- In the example, we could wind up with this

  \[
  \begin{array}{cccc}
  \text{Employee} & \text{ZipState} \\
  \hline
  \text{eid} & \text{address} & \text{city} & \text{zip} & \text{zip} & \text{state} \\
  123 & 111 1st Ave & Spokane & 99202 & 99203 & WA \\
  456 & 111 1st Ave & Spokane & 99258 & 99258 & WA \\
  \hline
  \end{array}
  \]

- We’ve lost the ability to enforce the FD: address, city, state \(\rightarrow\) zip
**Dependency Preserving Decompositions**

**Defining “dependency preserving”**

- Let $F$ be the FDs of a relation $R$
- Let $A$ and $B$ be sets of attributes in $R$

- An FD $X \rightarrow Y$ is “in $A$” if all attributes of $X$ and $Y$ are in $A$

- The “projection $F_A$” of $F$ on attributes $A$ are the FDs in $A$

- Assume $R$ is decomposed into sets of attributes $A$ and $B$
- The decomposition is “dependency preserving” if $(F_A \cup F_B)^+ = F^+$
  - Essentially, can we get $F$ back from just $F_A$ and $F_B$

**Example**

- $F = \{\text{address, city, state } \rightarrow \text{zip, zip } \rightarrow \text{state}\}$

Q: If $A = \{\text{eid, address, city, zip}\}$ and $B = \{\text{zip, state}\}$, what are $F_A$ and $F_B$?
  - $F_A = \emptyset$ neither FD is in $A$
  - $F_B = \{\text{zip } \rightarrow \text{state}\}$

- In this case, $(F_A \cup F_B)^+$ cannot equal $F^+$ since $F_B^+ \neq F^+$
- And so this decomposition is not dependency preserving!
Normalization goals (review)

Our goal for database design with FDs is:

- All relations in BCNF
- Only lossless decompositions
- Preserve dependencies

However:

- Dependency-preserving decomposition to BCNF not always possible
- But it is possible with 3NF (a weaker constraint than BCNF)

Definition of 3NF

- A schema \( R \) is in 3NF if for every FD \( X \rightarrow Y \), either:
  - \( X \rightarrow Y \) is a trivial FD (\( Y \subseteq X \)) from BCNF
  - \( X \rightarrow Y \) is a key FD (\( X \) is a superkey) from BCNF
  - \( Y \) is a part of some key for \( R \) not allowed in BCNF

The plan ...

- Go over how to decompose into 3NF
- Has an added benefit of sometimes giving a good BCNF decomposition
Canonical Covers and 3NF

Example

\[ \text{R}(a, b, c, d, e) \]

\[ F = \{ ab \rightarrow cde, a \rightarrow c, b \rightarrow d, d \rightarrow e, b \rightarrow e \} \]

We first have to compute a canonical cover \( F_c \) of \( F \)

- A set of dependencies “logically equivalent” to \( F \) such that:
  - no functional dependency in \( F_c \) contains an “extraneous” attribute
  - each left side of an FD is unique

Extraneous attributes

- if we can remove the attribute from a dependency in \( F \)
- without changing the closure of \( F \)

A cover for the example

- Start with:
  - \( ab \rightarrow cde \)
  - \( a \rightarrow c \)
  - \( b \rightarrow d \)
  - \( d \rightarrow e \)
  - \( b \rightarrow e \)
- Combine common left-hand sides:
  - \( ab \rightarrow cde \)
– a → c
– b → de
– d → e

• Remove extraneous attributes
  – ab → ∅
  – a → c
  – b → d
  – d → e

Apply 3NF decomposition algorithm
1. i = 0
2. For each FD X → Y in F_c
3. i = i + 1
4. R_i = X \cup Y
5. If none of the schemas contain a candidate key
6. i = i + 1
7. R_i = any candidate key
For our example, this gives the tables:

R1(a, c)
R2(b, d)
R3(d, e)
R4(a, b)

Q: Is this in BCNF?
   – Yes!
   – Sometimes the result is not only in 3NF, but also BCNF

This is an alternative approach:

• Decompose into 3NF
• Check if in BCNF
• If not, try to decompose problem relations into BCNF
• If the result is not in BCNF, revert to the 3NF decomposition
Conceptual Data Modeling Approaches

Similar to software design ...

- gather and analyse requirements
- design
- implement and test
- iterate

Design involves multiple steps prior to creating tables!

- We’ll focus on the **Entity-Relationship Model** (ER)
- Similar to UML diagrams in software design
- Sometimes called ER diagrams, or ERDs

![Diagram of entity-relationship model]

**Key:**
- Entity Set
- Relationship Set
- Attribute
ER Terminology

An “Entity”

• an object distinguishable from other objects
• e.g., the employee “John Smith”
• described using a set of attribute-value pairs
• one or more attributes as ids (keys)

An “Entity Set”

• a collection of similar entities
• defined by the kind of attributes and relationships the entities in the set are characterized by
• sometimes just called “Entity” (when it is clear we’re talking about the set)
• sometimes called an Entity Type
• an entity is an instance (or member) of an entity set
A **“Relationship”**

- an association among 2 or more entities
- e.g., John Smith's *home department* is Pharmacy 2

A **“Relationship Set”**

- a collection of similar relationships
- e.g., the set of *home department* relationships
- defined by the participating entity types and other constraints
- just called a “Relationship” (when context clear)
- also sometimes called a “Relationship Type”
- a relationship is *an instance* (or member) of a relationship set
ER vs. Relational Model

ER is a different data model than the relational model

- different *constructs* for modeling schemas and instances
- they are pretty close though

The *relational model* has ...

- tables (relations) with attributes, keys and foreign keys, rows, values

The *ER model* has ...

- entity and entity sets with attributes and entity identifiers (like keys)
- relationships and relationship sets with cardinality constraints, roles, attributes, etc.

Usually **map** ER models to relational schemas
ER Cardinality Constraints (Example)

- An Employee can have **0 or 1** home Departments
- A Department can have **0 or many** Employees
- A Department must have **exactly one** Manager
- etc.