Lecture 9:
Database Design

Wednesday, January 25, 2006

Closure of a set of Attributes

Given a set of attributes \( A_1, \ldots, A_n \)

The closure, \( \{A_1, \ldots, A_n\}^+ \), is the set of attributes \( B \) s.t. \( A_1, \ldots, A_n \rightarrow B \)

Example:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>\rightarrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>color</td>
</tr>
<tr>
<td>category</td>
<td>department</td>
</tr>
<tr>
<td>color, category</td>
<td>price</td>
</tr>
</tbody>
</table>

Closures:

- \( \text{name}^+ = \{\text{name, color}\} \)
- \( \{\text{name, category}\}^+ = \{\text{name, category, color, department, price}\} \)
- \( \text{color}^+ = \{\text{color}\} \)
Closure Algorithm

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

Repeat until $X$ doesn’t change do:

- If $B_1, \ldots, B_n \rightarrow C$ is a FD and $B_1, \ldots, B_n$ are all in $X$
  - then add $C$ to $X$.

Example:

$\{\text{name, category}\}^+ = \{\text{name, category, color, department, price}\}$

Hence: $\text{name, category} \rightarrow \text{color, department, price}$

Example

In class:

$R(A,B,C,D,E,F)$

\[
\begin{align*}
A, B & \rightarrow C \\
A, D & \rightarrow E \\
B & \rightarrow D \\
A, F & \rightarrow B 
\end{align*}
\]

Compute $\{A,B\}^+$ $X = \{A, B, \}$

Compute $\{A,F\}^+$ $X = \{A, F, \}$
Why Do We Need Closure

• With closure we can find all FD’s easily

• To check if $X \rightarrow A$
  – Compute $X^+$
  – Check if $A \in X^+$

Using Closure to Infer ALL FDs

Example:

\[
\begin{align*}
\text{A, B} & \rightarrow \text{C} \\
\text{A, D} & \rightarrow \text{B} \\
\text{B} & \rightarrow \text{D}
\end{align*}
\]

Step 1: Compute $X^+$, for every $X$:

\[
\begin{align*}
\text{A}^+ & = \text{A}, \quad \text{B}^+ = \text{BD}, \quad \text{C}^+ = \text{C}, \quad \text{D}^+ = \text{D} \\
\text{AB}^+ & = \text{ABCD}, \quad \text{AC}^+ = \text{AC}, \quad \text{AD}^+ = \text{ABCD} \\
\text{ABC}^+ & = \text{ABD}^+ = \text{ACD}^+ = \text{ABCD} \quad \text{(no need to compute– why ?)} \\
\text{BCD}^+ & = \text{BCD}, \quad \text{ABCD}^+ = \text{ABCD}
\end{align*}
\]

Step 2: Enumerate all FD’s $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

\[
\begin{align*}
\text{AB} & \rightarrow \text{CD}, \text{AD} \rightarrow \text{BC}, \quad \text{ABC} \rightarrow \text{D}, \text{ABD} \rightarrow \text{C}, \text{ACD} \rightarrow \text{B}
\end{align*}
\]
Another Example

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

What else can we infer? [in class, or at home]

Back to Conceptual Design

Now we know how to find more FDs, it’s easy
• Search for “bad” FDs
• If there are such, then decompose the table into two tables, repeat for the subtables.
• When done, the database schema is normalized

Unfortunately, there are several normal forms…
Normal Forms

First Normal Form = all attributes are atomic

Second Normal Form (2NF) = old and obsolete

Third Normal Form (3NF) = will discuss

Boyce Codd Normal Form (BCNF) = will discuss

Others...

Keys

• A superkey is a set of attributes $A_1, ..., A_n$ s.t. for any other attribute $B$, we have $A_1, ..., A_n \rightarrow B$

• A key is a minimal superkey
  – I.e. set of attributes which is a superkey and for which no subset is a superkey
Computing (Super)Keys

- Compute $X^+$ for all sets $X$
- If $X^+ = \text{all attributes}$, then $X$ is a key
- List only the minimal $X$’s

Example

Product(name, price, category, color)

- name, category $\rightarrow$ price
- category $\rightarrow$ color

What is the key?
Example

Product(name, price, category, color)

\[
\begin{align*}
\text{name, category} & \rightarrow \text{price} \\
\text{category} & \rightarrow \text{color}
\end{align*}
\]

What is the key?

\[(\text{name, category}) + = \text{name, category, price, color}\]

Hence (name, category) is a key

Examples of Keys

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

\[
\begin{align*}
\text{student} & \rightarrow \text{address} \\
\text{room, time} & \rightarrow \text{course} \\
\text{student, course} & \rightarrow \text{room, time}
\end{align*}
\]

(find keys at home)
Eliminating Anomalies

Main idea:

• $X \rightarrow A$ is OK if $X$ is a (super)key

• $X \rightarrow A$ is not OK otherwise

Example

What the key?

{SSN, PhoneNumber}

Hence SSN $\rightarrow$ Name, City is a “bad” dependency
Key or Keys?

Can we have more than one key?

Given R(A,B,C) define FD’s s.t. there are two or more keys

AB → C
BC → A

or

A → BC
B → AC

What are the keys here?
Can you design FDs such that there are three keys?
Boyce-Codd Normal Form

A simple condition for removing anomalies from relations:

A relation R is in BCNF if:

- If $A_1, ..., A_n \rightarrow B$ is a non-trivial dependency in R, then $\{A_1, ..., A_n\}$ is a superkey for R

In other words: there are no “bad” FDs

Equivalently:

$\forall X$, either $(X^+ = X)$ or $(X^+ = \text{all attributes})$

BCNF Decomposition Algorithm

repeat

- choose $A_1, ..., A_m \rightarrow B_1, ..., B_n$ that violates BCNF
- split R into $R_1(A_1, ..., A_m, B_1, ..., B_n)$ and $R_2(A_1, ..., A_m, \text{[others]})$
- continue with both $R_1$ and $R_2$

until no more violations

Is there a 2-attribute relation that is not in BCNF?

In practice, we have a better algorithm (coming up)
Example

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-1234</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

SSN → Name, City

What the key?

{SSN, PhoneNumber} use SSN → Name, City to split

Example

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

SSN → Name, City

Let’s check anomalies:
- Redundancy?
- Update?
- Delete?
**Example Decomposition**

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

SSN $\rightarrow$ name, age
age $\rightarrow$ hairColor

Decompose in BCNF (in class):

---

**BCNF Decomposition Algorithm**

BCNF\_Decompose(R)

find X s.t.: $X \neq X^+ \neq$ [all attributes]

**if** (not found) **then** “R is in BCNF”

**let** $Y = X^+ - X$

**let** $Z = [\text{all attributes}] - X^+$

decompose R into $R_1(X \cup Y)$ and $R_2(X \cup Z)$

continue to decompose recursively $R_1$ and $R_2$
Example BCNF Decomposition

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

SSN → name, age
age → hairColor

Iteration 1: Person
SSN+ = SSN, name, age, hairColor
Decompose into: P(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)

Iteration 2: P
age+ = age, hairColor
Decompose: People(SSN, name, age)
Hair(age, hairColor)
Phone(SSN, phoneNumber)

Find X s.t.: X ≠ X+ ≠ [all attributes]

Example

R(A,B,C,D)

A → B
B → C

R(A,B,C,D)
A+ = ABC ≠ ABCD

R1(A,B,C)
B+ = BC ≠ ABC

R11(B,C)
R12(A,B)

R2(A,D)

What are the keys?

What happens if in R we first pick B+? Or AB2+?
Decompositions in General

\[ R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p) \]

\[ R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]

\[ R_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

\( R_1 = \text{projection of } R \text{ on } A_1, \ldots, A_n, B_1, \ldots, B_m \)

\( R_2 = \text{projection of } R \text{ on } A_1, \ldots, A_n, C_1, \ldots, C_p \)

Theory of Decomposition

- Sometimes it is correct:

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

Lossless decomposition
Incorrect Decomposition

- Sometimes it is not:

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

Lossy decomposition

Decompositions in General

\[ R(A_1, ..., A_n, B_1, ..., B_m, C_1, ..., C_p) \]

\[ R_1(A_1, ..., A_n, B_1, ..., B_m) \]

\[ R_2(A_1, ..., A_n, C_1, ..., C_p) \]

If \( A_1, ..., A_n \rightarrow B_1, ..., B_m \), Then the decomposition is lossless

Note: don’t need \( A_1, ..., A_n \rightarrow C_1, ..., C_p \)

BCNF decomposition is always lossless. WHY?
3NF: A Problem with BCNF

We lose the FD: `Company, Product → Unit` !

So What’s the Problem?

No problem so far. All `local` FD’s are satisfied. Let’s put all the data back into a single table again:

Violates the FD: `Company, Product → Unit`
The Problem

• We started with a table R and FD

• We decomposed R into BCNF tables \( R_1, R_2, \ldots \) with their own \( FD_1, FD_2, \ldots \)

• We can reconstruct R from \( R_1, R_2, \ldots \)

• But we cannot reconstruct FD from \( FD_1, FD_2, \ldots \)

Solution: 3rd Normal Form (3NF)

A simple condition for removing anomalies from relations:

A relation R is in 3rd normal form if:

Whenever there is a nontrivial dependency \( A_1, A_2, \ldots, A_n \rightarrow B \) for \( R \), then \( \{A_1, A_2, \ldots, A_n\} \) a super-key for \( R \), or B is part of a key.

Tradeoff:

BCNF = no anomalies, but may lose some FDs
3NF = keeps all FDs, but may have some anomalies
3NF Decomposition Algorithm

3NF_Decompose(R)

let K = [all attributes that are part of some key]

find X s.t.: X+ - X - K ≠ Ø and X+ ≠ [all attributes]

if (not found) then “R is already in 3NF”

let Y = X+ - X - K

let Z = [all attributes] - (X ∪ Y)

decompose into R1(X ∪ Y) and R2(X ∪ Z)

decompose, recursively, R1 and R2

Example of 3NF decomposition

R(A,B,C,D,E):

<table>
<thead>
<tr>
<th>AB → C</th>
<th>C → D</th>
<th>D → B</th>
<th>D → E</th>
</tr>
</thead>
</table>

Keys: (need to compute X+, for several Xs)

AB, AC, AD

K = {A, B, C, D}

Pick X = C
C+ = BCDE
C → BDE is a BCNF violation
For 3NF: remove B, D (part of K):
C → E is a 3NF violation
Decompose: R1(C, E), R2(A,B,C,D)

R1 is in 3NF
R2 is in 3NF (because its keys: AB, AC, AD)
FD’s for E/R Diagrams

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

**Rule 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.

- Person
  - address
  - name
  - ssn

Person(address, name, ssn)
FD’s for E/R Diagrams

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

```
Person
  name
  ssn

Product
  name
  price

buys
  date

buys(name, ssn, date)
```

FD’s for E/R Diagrams

**Except:** if there is an arrow from the relationship to E, then we don’t need the key of E as part of the relation key.

```
Person
  name
  ssn

CreditCard
  name
  card-no

Purchase
  surname

Store

Purchase(name, surname, ssn, card-no)
```
FD’s for E/R Diagrams

More rules:
• Many-one, one-many, one-one relationships
• Multi-way relationships
• Weak entity sets

(Try to find them yourself, or check book)