DECOMPOSITION & SCHEMA NORMALIZATION

CS 564- Spring 2018

ACKs: Dan Suciu, Jignesh Patel, AnHai Doan
WHAT IS THIS LECTURE ABOUT?

• Bad schemas lead to redundancy
• To “correct” bad schemas: decompose relations
  – lossless-join
  – dependency preserving
• Desired normal forms
  – BCNF
  – 3NF
DB DESIGN THEORY

• Helps us identify the “bad” schemas and improve them
  1. express constraints on the data: functional dependencies (FDs)
  2. use the FDs to decompose the relations

• The process, called normalization, obtains a schema in a “normal form” that guarantees certain properties
  – examples of normal forms: BCNF, 3NF, ...
Schema Decomposition
WHAT IS A DECOMPOSITION?

We decompose a relation $R(A_1, ..., A_n)$ by creating

- $R_1(B_1, .., B_m)$
- $R_2(C_1, ..., C_l)$
- where $\{B_1, ..., B_m\} \cup \{C_1, ..., C_l\} = \{A_1, ..., A_n\}$

- The instance of $R_1$ is the projection of $R$ onto $B_1, .., B_m$
- The instance of $R_2$ is the projection of $R$ onto $C_1, .., C_l$
EXAMPLE: DECOMPOSITION

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

CS 564 [Spring 2018] - Paris Koutris 6
What should a good decomposition achieve?

1. minimize redundancy
2. avoid information loss (lossless-join)
3. preserve the FDs (dependency preserving)
4. ensure good query performance
# EXAMPLE: INFORMATION LOSS

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>24</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>John</td>
<td>24</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>Arun</td>
<td>20</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

Decompose into:

- $R_1(\text{name, age})$
- $R_2(\text{age, phoneNumber})$

We can’t figure out which phoneNumber corresponds to which person!
A schema decomposition is **lossless-join** if for any initial instance $R, R = R'$.
A LOSSLESS-JOIN CRITERION

Starting with:
• a relation \( R(A) \) + set \( F \) of FDs
• a decomposition of \( R \) into \( R_1(A_1) \) and \( R_2(A_2) \)

we say that a decomposition is lossless-join if and only if at least one of the following FDs is in \( F^+ \) (the closure of \( F \)):

1. \( A_1 \cap A_2 \rightarrow A_1 \)
2. \( A_1 \cap A_2 \rightarrow A_2 \)
EXAMPLE

- relation \( R(A, B, C, D) \)
- FD \( A \rightarrow B, C \)

Lossless-join
- decomposition into \( R_1(A, B, C) \) and \( R_2(A, D) \)
- \( \{A, B, C\} \cap \{A, D\} = \{A\} \)
- For \( R_1 \) we have indeed \( A \rightarrow B, C \)

Not lossless-join
- decomposition into \( R_1(A, B, C) \) and \( R_2(D) \)
Given \( R \) and a set of FDs \( F \), we decompose \( R \) into \( R_1 \) and \( R_2 \). Suppose:

- \( R_1 \) has a set of FDs \( F_1 \)
- \( R_2 \) has a set of FDs \( F_2 \)
- \( F_1 \) and \( F_2 \) are computed from \( F \)

A decomposition is \textit{dependency preserving} if by enforcing \( F_1 \) over \( R_1 \) and \( F_2 \) over \( R_2 \), we can enforce \( F \) over \( R \)
GOOD EXAMPLE

Person(SSN, name, age, canDrink)

• \( SSN \rightarrow name, age \)
• \( age \rightarrow canDrink \)

decomposes into

• \( R_1(SSN, name, age) \)
  – \( SSN \rightarrow name, age \)
• \( R_2(age, canDrink) \)
  – \( age \rightarrow canDrink \)
BAD EXAMPLE

\( R(A, B, C) \)
- \( A \rightarrow B \)
- \( B, C \rightarrow A \)

Decomposes into:
- \( R_1(A, B) \)
  - \( A \rightarrow B \)
- \( R_2(A, C) \)
  - no FDs here!!

\[ \begin{array}{|c|c|}
\hline
A & B \\
\hline
a_1 & b \\
\hline
a_2 & b \\
\hline
\end{array} \]

\[ \begin{array}{|c|c|}
\hline
A & C \\
\hline
a_1 & c \\
\hline
a_2 & c \\
\hline
\end{array} \]

The recovered table violates \( B, C \rightarrow A \).
NORMAL FORMS

A normal form represents a “good” schema design:

- 1NF (flat tables/atomic values)
- 2NF
- 3NF
- BCNF
- 4NF
- ...

more restrictive
BCNF Decomposition
A relation $R$ is in **BCNF** if whenever $X \rightarrow B$ is a non-trivial FD, then $X$ is a **superkey** in $R$.

**Equivalent definition**: for every attribute set $X$

- either $X^+ = X$
- or $X^+ = \text{all attributes}$
**BCNF Example 1**

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

\[ SSN \rightarrow name, age \]

- **key** = \( \{ SSN, phoneNumber \} \)
- \( SSN \rightarrow name, age \) is a “bad” FD
- The above relation is **not** in BCNF!
**BCNF EXAMPLE 2**

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
</tr>
</tbody>
</table>

\[ SSN \rightarrow \text{name, age} \]

- **key** = \{SSN\}
- The above relation is in BCNF!
BCNF Example 3

<table>
<thead>
<tr>
<th>SSN</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

- **key** = \{SSN, phoneNumber\}
- The above relation is in BCNF!
- Is it possible that a binary relation is not in BCNF?
BCNF DECOMPOSITION

• Find an FD that violates the BCNF condition

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

• Decompose \( R \) to \( R_1 \) and \( R_2 \):

• Continue until no BCNF violations are left

\( R_1 \) \( R_2 \)

B’s \( \text{remaining attributes} \) A’s
EXAMPLE

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

- The FD $SSN \rightarrow name, age$ violates BCNF
- Split into two relations $R_1$, $R_2$ as follows:
EXAMPLE CONT’D

\[ \text{SSN} \rightarrow \text{name, age} \]

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>
BCNF DECOMPOSITION PROPERTIES

The BCNF decomposition:

– removes certain types of redundancy
– is lossless-join
– is not always dependency preserving
**BCNF IS LOSSLESS-JOIN**

**Example:**

\[ R(A, B, C) \text{ with } A \rightarrow B \text{ decomposes into: } R_1(A, B) \text{ and } R_2(A, C) \]

- The BCNF decomposition always satisfies the lossless-join criterion!
The BCNF decomposition is:

- $R_1(A, B)$ with FD $A \rightarrow B$
- $R_2(A, C)$ with no FDs
BCNF EXAMPLE (1)

Books (author, gender, booktitle, genre, price)
- author → gender
- booktitle → genre, price

What is the candidate key?
- (author, booktitle) is the only one!

Is is in BCNF?
- No, because the left hand side of both (not trivial) FDs is not a superkey!
BCNF Example (2)

Books (author, gender, booktitle, genre, price)
• author → gender
• booktitle → genre, price

Splitting Books using the FD author → gender:
• Author (author, gender)
  FD: author → gender in BCNF!
• Books2 (authos, booktitle, genre, price)
  FD: booktitle → genre, price not in BCNF!
**BCNF EXAMPLE (3)**

**Books** (author, gender, booktitle, genre, price)

- *author* $\rightarrow$ *gender*
- *booktitle* $\rightarrow$ *genre, price*

Splitting **Books** using the FD *author* $\rightarrow$ *gender*:

- **Author** (author, gender)
  
  FD: *author* $\rightarrow$ *gender* in BCNF!

- Splitting **Books2** (author, booktitle, genre, price):
  - **BookInfo** (booktitle, genre, price)
    
    FD: *booktitle* $\rightarrow$ *genre, price* in BCNF!
  - **BookAuthor** (author, booktitle) in BCNF!
THIRD NORMAL FORM (3NF)
A relation $R$ is in **3NF** if whenever $X \rightarrow A$, one of the following is true:

- $A \in X$ (trivial FD)
- $X$ is a superkey
- $A$ is part of some key of $R$ (prime attribute)

BCNF implies 3NF !!
3NF CONT’D

- **Example**: $R(A, B, C)$ with $A, B \rightarrow C$ and $C \rightarrow A$
  - is in 3NF. Why?
  - is not in BCNF. Why?

- Compromise used when BCNF not achievable: *aim for BCNF and settle for 3NF*

- Lossless-join and dependency preserving decomposition into a collection of 3NF relations is always possible!
3NF ALGORITHM

1. Apply the algorithm for BCNF decomposition until all relations are in 3NF (we can stop earlier than BCNF)
2. Compute a minimal basis $F'$ of $F$
3. For each non-preserved FD $X \rightarrow A$ in $F'$, add a new relation $R(X, A)$
3NF EXAMPLE (1)

Start with relation \( R \) (A, B, C, D) with FDs:

- \( A \rightarrow D \)
- \( A, B \rightarrow C \)
- \( A, D \rightarrow C \)
- \( B \rightarrow C \)
- \( D \rightarrow A, B \)

**Step 1:** find a BCNF decomposition

- \( R1 \) (B, C)
- \( R2 \) (A, B, D)
3NF EXAMPLE (2)

Start with relation $R$ (A, B, C, D) with FDs:

- $A \rightarrow D$
- $A, B \rightarrow C$
- $A, D \rightarrow C$
- $B \rightarrow C$
- $D \rightarrow A, B$

**Step 2**: compute a minimal basis of the original set of FDs:

- $A \rightarrow D$
- $B \rightarrow C$
- $D \rightarrow A$
- $D \rightarrow B$
Start with relation $R$ $(A, B, C, D)$ with FDs:

- $A \rightarrow D$
- $A, B \rightarrow C$
- $A, D \rightarrow C$
- $B \rightarrow C$
- $D \rightarrow A, B$

**Step 3**: add a new relation for any FD in the basis that is not satisfied:

- all the dependencies in $F'$ are satisfied!
- the resulting decomposition $R_1, R_2$ is also BCNF!
IS NORMALIZATION ALWAYS GOOD?

• **Example**: suppose A and B are always used together, but normalization says they should be in different tables
  – decomposition might produce unacceptable performance loss

• **Example**: data warehouses
  – huge historical DBs, rarely updated after creation
  – joins expensive or impractical