CS 133: Databases

Fall 2018
Lec 22 – 11/27
Database Design
Prof. Beth Trushkowsky

Rules of Inference

• **Armstrong’s Axioms** \( (X, Y, Z \text{ are sets of attributes}) \):
  
  – **Reflexivity**: If \( Y \subseteq X \), then \( X \rightarrow Y \)
  
  – **Augmentation**: If \( X \rightarrow Y \), then \( XZ \rightarrow YZ \) for any \( Z \)
  
  – **Transitivity**: If \( X \rightarrow Y \) and \( Y \rightarrow Z \), then \( X \rightarrow Z \)

• Some additional rules (that follow from AA):
  
  – **Union**: If \( X \rightarrow Y \) and \( X \rightarrow Z \), then \( X \rightarrow YZ \)
  
  – **Decomposition**: If \( X \rightarrow YZ \), then \( X \rightarrow Y \) and \( X \rightarrow Z \)
  
  – **Pseudo-transitivity**: If \( X \rightarrow Y \) and \( YW \rightarrow Z \), then \( XW \rightarrow Z \)

Goals for Today

• Learn how to decompose a relation to adhere to Boyce-Codd Normal Form (BCNF)

• Understand lossy vs. loss-less decompositions

• Reason about issues that can result even if a decomposition is loss-less

The Issue with Non-Key FDs

• Why does the FD \( \text{rating} \rightarrow \text{hourly\_wages} \) yield redundancy issues?

  • **Rating** is a **non-key field**, so there could be **duplicate pairs** of particular \( \{\text{rating}, \text{hourly\_wages}\} \) in this relation

  • By separating \( \{\text{rating}, \text{hourly\_wages}\} \) into its own relation, we resolve redundancy!
    – Can regain the original data via **natural join**
“Normal” Forms for a Schema

- **Idea:** decompose relation into two or more relations to remove redundancy. Decomposition **guided by FDs**!

- **Boyce-Codd Normal Form (BCNF)**
  - Adhere to simple conditions and anomalies caused by data redundancy cannot occur

- BCNF definition:
  A Relation R with FDs F is in BCNF if, for all \( X \rightarrow A \) in \( F^+ \)
  - \( A \in X \) (a trivial FD), or
  - \( X \) is a superkey for R

- I.e.,: R is in BCNF if the **only non-trivial FDs over** R are **key constraints**

Lossy vs. Lossless Decomposition

- Example schema:
  Oversees(\text{ProjectId, EmployeeId, DepartmentId})

- FDs:
  - \( E \rightarrow P \) (an employee oversees only one project)
  - \( D \rightarrow P \) (a dept works on only one project)
  - \( E \rightarrow D \) (an employee only works with one dept for these projects)

- Example instance of Oversees:

<table>
<thead>
<tr>
<th>Project</th>
<th>Employee</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Alice</td>
<td>Physics</td>
</tr>
<tr>
<td>Comet</td>
<td>Bob</td>
<td>Astronomy</td>
</tr>
<tr>
<td>Genomics</td>
<td>Carl</td>
<td>Biology</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
<td>Biology</td>
</tr>
</tbody>
</table>

**Redundancy?**

Problems with Decompositions

- There are three potential problems to consider:
  1. May be **impossible to reconstruct the original relation**! (**Lossiness**)  
  2. Checking functional dependencies may require joins
  3. Some queries become more expensive due to joins
    - e.g., *How much does Smiley earn?*

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Atthisoo</td>
<td>48</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

**Wages**

<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

*E.g.: Hourly_Emps2*

Lossy vs. Lossless Decomp (cntd)

- **Redundancy with the FD** \( D \rightarrow P \)

<table>
<thead>
<tr>
<th>Project</th>
<th>Employee</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Alice</td>
<td>Physics</td>
</tr>
<tr>
<td>Comet</td>
<td>Bob</td>
<td>Astronomy</td>
</tr>
<tr>
<td>Genomics</td>
<td>Carl</td>
<td>Biology</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
<td>Biology</td>
</tr>
</tbody>
</table>

- **Proposed decomposition:**

<table>
<thead>
<tr>
<th>Project</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Alice</td>
</tr>
<tr>
<td>Comet</td>
<td>Bob</td>
</tr>
<tr>
<td>Genomics</td>
<td>Carl</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
</tr>
</tbody>
</table>
Lossy vs. Lossless Decomp (cntd)

• Redundancy with the FD $D \rightarrow P$

<table>
<thead>
<tr>
<th>Project</th>
<th>Employee</th>
<th>Department</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Alice</td>
<td>Physics</td>
<td>Comet</td>
</tr>
<tr>
<td>Comet</td>
<td>Bob</td>
<td>Astronomy</td>
<td>Comet</td>
</tr>
<tr>
<td>Genomics</td>
<td>Carl</td>
<td>Biology</td>
<td>Genomics</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
<td>Biology</td>
<td>Genomics</td>
</tr>
</tbody>
</table>

Tuples not in original!

Lossy vs. Lossless Decomp (cntd)

• Decomposition attempt #2, for FD $D \rightarrow P$:

<table>
<thead>
<tr>
<th>Department</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics</td>
<td>Alice</td>
</tr>
<tr>
<td>Astronomy</td>
<td>Bob</td>
</tr>
<tr>
<td>Biology</td>
<td>Carl</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
</tr>
</tbody>
</table>

Loss-less Decomposition

• Decomposition of $R$ into $X$ and $Y$ is lossless-join w.r.t. a set of FDs $F$ if, for every instance $r$ that satisfies $F$:

$$\pi_X(r) \supseteq \pi_Y(r) = r$$

• Decomposition of $R$ into $X$ and $Y$ is lossless with respect to $F$ if and only if $F^+$ contains:

- $X \cap Y \rightarrow X$, or
- $X \cap Y \rightarrow Y$

**In other words, the common attributes form a key for $X$ or $Y$**

**Corollary:** If $Z \rightarrow W$ holds over $R$ and $Z \cap W$ is empty, then decomposition of $R$ into $ZW$ and $R-W$ is loss-less.

• In “Oversees” example, decomposing into $\{E,P\}$ and $\{D,P\}$ is lossy because the intersection (i.e., $Project$) is not a key of either resulting relation

Loss-less Decomposition into BCNF

• Relation $R$ has FDs $F$. If $Z \rightarrow W$ in $F$ violates BCNF:
  – decompose $R$ into $R-W$ and $ZW$ (guaranteed to be loss-less)
Reasoning about BCNF

- Relation R with FDs F is in BCNF if, for all X → A in F⁺
  - A ∈ X (a trivial FD), or
  - X is a superkey for R

Also recall that relations are sets of tuples

Example 1: Is Hourly_Emps in BCNF?

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>W</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>7</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldru</td>
<td>35</td>
<td>5</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

- SNLRWH has FDs
  S → SNLRWH
  R → W

Example 2: Is Bar_Sells in BCNF?

- Combing Bars and Sells
  Bar_Sells (bar_name, beer_name, address, price)

- FDs (for just Bar_Sells):
  bar_name → address
  bar_name, beer_name → price

In BCNF??

(Exercise 2)
Examples: BCNF Decomposition

• **Hourly_Emps**

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>231-31-368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

**Transitive Dependencies**

• Violating FD involves *attribute(s) depending on non-key attribute(s)*

• **Bar_sells**

Partial Dependencies

• Violating FD involves *attribute(s) depending on attribute(s) that are proper subset of a key*

Key = {beer_name, bar_name}

Repeated Decomposition

• *Repeated decomposition*
  – May be needed to get set of relations that are in **BCNF**
  – Can confirm BCNF for original relation R using only FDs F, but each decomposed relation R_i must be checked for violating each [relevant] FD in F^+

• Using *attribute closure* to check decomposed R_i
  – To confirm R_i is in BCNF: for each subset of attributes α in R_i, check that α^+ (under F):
    • Contains no attributes of R_i − α, or
    • Contains all attributes of R_i

**Bar_sells** violating FD: bar_name → address
Exercise 3: BCNF Decomposition

- Candidate key = \{id, advisorId\}
- FD violation? Both!
- Decomposed into three relations:
  - R1 = \{id, name, dorm\}
  - R2 = \{advisorId, advisorName\}
  - R3 = \{id, advisorId\}

An Aside: *Multiple* Candidate Keys

- For relation $\text{Bars}(\text{bar\_name}, \text{address})$, suppose we knew:
  - bar\_name $\rightarrow$ address
  - address $\rightarrow$ bar\_name
  Either attribute could serve as primary key!

- When creating a relation in SQL, use *one* candidate key as the primary key
  - Enforce others using UNIQUE key word
  - Commonly used when use *surrogate key* as a primary key

Dependency Preservation

- Decomposed example from “Oversees”:
  - E $\rightarrow$ P (an employee oversees only one project)
  - D $\rightarrow$ P (a dept works on only one project)
  - E $\rightarrow$ D (an employee only works with one dept for these projects)

<table>
<thead>
<tr>
<th>Project</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Physics</td>
</tr>
<tr>
<td>Comet</td>
<td>Astronomy</td>
</tr>
<tr>
<td>Genomics</td>
<td>Biology</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Department</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics</td>
<td>Alice</td>
</tr>
<tr>
<td>Astronomy</td>
<td>Bob</td>
</tr>
<tr>
<td>Biology</td>
<td>Carl</td>
</tr>
<tr>
<td>Biology</td>
<td>Denise</td>
</tr>
</tbody>
</table>

- How can we check E $\rightarrow$ P ??
  (an employee oversees only one project)

Dependency Preserving Decomposition

- **Dependency preserving decomposition** (Intuition):
  - If R is decomposed into X, Y and Z, and we enforce the FDs that hold individually on X, Y, and Z
    $\rightarrow$ then all FDs that were given to hold on R must also hold

- The projection of F on attribute set X (denoted $F_X$):
  - The set of FDs $U \rightarrow V$ in $F^+$ (closure of F, not just F!) such that all of the attributes on both sides of the FD are in X
  - That is: $U$ and $V$ are subsets of X
**Dependency Preserving Decompositions (Contd.)**

- Decomposition of R into X and Y is **dependency preserving** if \((F_X \cup F_Y)^+ = F^+\)
  - i.e., we can check FDs on X and Y independently

- “Oversees” example, continued:
  - \(X = \{\text{Employee, Department}\}, \ F_X = \{\ E \rightarrow D \}\)
  - \(Y = \{\text{Project, Department}\}, \ F_Y = \{\ D \rightarrow P \}\)
  - Does \((F_X \cup F_Y)^+\) include \(E \rightarrow P\)?
    
    **YES!** (transitive property)

---

**Movie showings: decomposition issue**

- Showings (movie, theater, city)
  - FDs
    - movie, city --> theater
    - theater --> city

**Violating FD!**

Decompose...

**Above decomposition could allow this to happen!**

**Violates FD**

movie, city --> theater

**Third Normal Form (3NF)**

- **Definition:** for all \(X \rightarrow A\) in \(F^+\)
  - \(A \in X\) (called a trivial FD), or
  - \(X\) is a superkey for \(R\), **OR**
  - \(A\) is a part of some candidate key for \(R\)
- Allows FDs like non-key \(\rightarrow\) partial key

- 3NF but not BCNF?
  - have overlapping composite candidate keys

**Always possible to get a loss-less, dependency-preserving decomposition into 3NF!**

(may contain redundancy)

---

**Alternate Formulation of 3NF & BCNF**

Every non-key attribute must describes a fact about “the key, the whole key, and nothing but the key, so help me Codd”

- Normal forms increasingly restrictive
  - 1st NF ⊇ 2nd NF ⊇ 3rd NF ⊇ Boyce-Codd NF