Today

- BCNF

Assignments

- HW 8 out, due Tues
- PROJ 1 out, due Tues
Boyce-Codd Normal Form (BCNF)

A relation is in BCNF if all of its FDs are either

- **Trivial FDs** (e.g., $AB \rightarrow A$)

  or

- **Key FDs** (of the form $A \rightarrow B$ for a key $A$)

Are any of these relations are in BCNF? (Why or why not...)

- EmpDept(eid, name, dept, dept_name)
- Assigned(eid, pid, emp_name, percent)
- Enrollment(sid, cid, grade, instructor, student_name, instr_name)

BCNF relations have no redundancy caused by FDs

- redundancy if there is an FD between attributes
- and there can be repeated entries of data for those attributes
- for example:

<table>
<thead>
<tr>
<th>dept</th>
<th>dept_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>CS</td>
</tr>
<tr>
<td>10</td>
<td>HR</td>
</tr>
<tr>
<td>12</td>
<td>CS</td>
</tr>
</tbody>
</table>

- if a relation is in BCNF, the FD must be a key FD
- in the example dept must be a key
- which implies, e.g., $(12, CS)$ can only appear once
Decomposition into BCNF

A BCNF decomposition algorithm for a relation $R$ (assuming $F^+$)

1. Let $A$ be the set of attributes of $R$
2. If $R$ is not in BCNF and $X \rightarrow Y$ is a non-trivial, non-key FD of $R$
3. Decompose $R$ into $A - Y$ and $XY$
4. If $A - Y$ and/or $XY$ is not in BCNF
5. recursively apply step 1 (to $A - Y$ and/or $XY$)

Exercise

Q: Decompose the following into BCNF relations ...

Q: Write an SQL query to “get back” the original table ...

    Enrollment(sid, cid, grade, instructor, student_name)

- Start with the non-key FD: sid $\rightarrow$ student_name
- This gives the decomposition:

    Enrollment(sid, cid, grade, instructor)
    Student(sid, student_name)

- Now use the non-key FD: cid $\rightarrow$ instructor
- This gives the decomposition:

    Enrollment(sid, cid, grade)
    Student(sid, student_name)
    Instructor(cid, instructor)

- All relations are now in BCNF!
Where are we?

We’ve accomplished a lot!

- began with a relational schema
- identified problems with redundancy (anomalies)
- used FDs to eliminate those problems via decompositions to BCNF
- on the way learned how to identify keys using FDs

There are two steps left...

- ensure the BCNF decompositions do not lose information ("lossy")
- in some cases we may lose FDs, but there are ways to deal with this
Lossless Decomposition

- Assume a schema is decomposed into two attribute sets \( A_1 \) and \( A_2 \)
- A decomposition is “\textbf{lossless}” if for every instance \( R \) of the original schema

\[
R = \text{SELECT } * \\
\text{FROM (SELECT } A_1 \text{ FROM } R) \\
\text{NATURAL JOIN} \\
(\text{SELECT } A_2 \text{ FROM } R)
\]

- That is, we can recover \( R \) from the natural join of the decomposition

Example

Enrollment

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>instructor</th>
<th>student_name</th>
<th>instructor_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>001</td>
<td>1</td>
<td>alice</td>
<td>bob</td>
</tr>
<tr>
<td>246</td>
<td>002</td>
<td>2</td>
<td>charlie</td>
<td>diane</td>
</tr>
</tbody>
</table>

Q: Is \( A_1 = \{ \text{sid, student\_name} \} \) and \( A_2 = \{ \text{cid, instructor, instructor\_name} \} \) a lossless decomposition?

- No!
- The \textsc{NATURAL JOIN} is just the \textsc{CROSS JOIN}

Q: How can we change this decomposition to make it lossless?

- In this case, need to leave the key in one of the tables
Generating Lossless Decompositions

- Decomposing a table into attribute sets $A_1$ and $A_2$ is lossless iff $A_1 \cap A_2$ contains a key for either $A_1$ or $A_2$
  - if they have a key in common, they can be joined back together
  - $\{\text{sid, student name}\} \cap \{\text{cid, instructor, instructor name}\} = \emptyset$

Q: Does the BCNF algorithm guarantee lossless decompositions?

- Yes!
  - Assume $A$ is the set of attributes
  - For FD $X \rightarrow Y$, we have: $A_1 = A - Y$ and $A_2 = X \cup Y$
  - Thus $A_1 \cap A_2 = X$ and $X$ is a key for $X \cup Y$
Dependency-Preserving Decompositions

Decompositions should also **preserve dependencies**

- For example:

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

  - \( \text{zip} \to \text{state} \) (note not totally accurate)
  - \( \text{address}, \text{city}, \text{state} \to \text{zip} \)

- 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:

  - \( \text{address}, \text{city}, \text{state} \to \text{zip} \)

- In the example, we could wind up with this

<table>
<thead>
<tr>
<th>Employee</th>
<th>ZipState</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{eid}</td>
<td>\text{address}</td>
</tr>
<tr>
<td>123</td>
<td>111 1st Ave</td>
</tr>
<tr>
<td>456</td>
<td>111 1st Ave</td>
</tr>
<tr>
<td>\text{zip}</td>
<td>\text{state}</td>
</tr>
<tr>
<td>99203</td>
<td>WA</td>
</tr>
<tr>
<td>99258</td>
<td>WA</td>
</tr>
</tbody>
</table>

- We’ve lost the ability to enforce the FD: \( \text{address}, \text{city}, \text{state} \to \text{zip} \)
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

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