Relational Database Design Theory
Part I

CPS 116
Introduction to Database Systems

Announcements (September 12)
- Homework #1 due next Tuesday
- Help session this Wednesday
  - 4:30pm or 5:30pm?
  - D344 LSRC
  - Email reminder tonight
- Course project assigned today
  - Choice of “standard” or “open”
  - Milestone 1 right after fall break
    - But plan/start early!!!

Motivation

- How do we tell if a design is bad, e.g., $\text{StudentEnroll}(\text{SID}, \text{name}, \text{CID})$?
  - This design has redundancy, because the name of a student is recorded multiple times, once for each course the student is taking
- How about a systematic approach to detecting and removing redundancy in designs?
  - Dependencies, decompositions, and normal forms

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>CID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Jane</td>
<td>CPS116</td>
</tr>
<tr>
<td>14</td>
<td>Bart</td>
<td>CPS116</td>
</tr>
<tr>
<td>37</td>
<td>Lisa</td>
<td>CPS116</td>
</tr>
<tr>
<td>37</td>
<td>Lisa</td>
<td>CPS130</td>
</tr>
</tbody>
</table>

Functional dependencies

- A functional dependency (FD) has the form $X \rightarrow Y$, where $X$ and $Y$ are sets of attributes in a relation $R$
- $X \rightarrow Y$ means that whenever two tuples in $R$ agree on all the attributes in $X$, they must also agree on all attributes in $Y$

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>?</td>
</tr>
</tbody>
</table>

- Must be $\delta$ Could be anything

FD examples

$\text{Address}(\text{street_address}, \text{city}, \text{state}, \text{zip})$
- $\text{street_address}, \text{city}, \text{state} \rightarrow \text{zip}$
- $\text{zip} \rightarrow \text{city}, \text{state}$
- $\text{zip}, \text{state} \rightarrow \text{zip}$
  - This is a trivial FD
  - Trivial FD: LHS $\supset$ RHS
- $\text{zip} \rightarrow \text{state}, \text{zip}$
  - This is non-trivial, but not completely non-trivial
  - Completely non-trivial FD: LHS $\cap$ RHS = $\emptyset$

Keys redefined using FD’s

A set of attributes $K$ is a key for a relation $R$ if
- $K \rightarrow$ all (other) attributes of $R$
  - That is, $K$ is a “super key”
- No proper subset of $K$ satisfies the above condition
  - That is, $K$ is minimal
Reasoning with FD’s

Given a relation \( R \) and a set of FD’s \( F \)

- Does another FD follow from \( F \)?
  - Are some of the FD’s in \( F \) redundant (i.e., they follow from the others)?
- Is \( K \) a key of \( R \)?
  - What are all the keys of \( R \)?

Attribute closure

- Given \( R \), a set of FD’s \( F \) that hold in \( R \), and a set of attributes \( Z \) in \( R \):
  - The closure of \( Z \) (denoted \( Z^+ \)) with respect to \( F \) is the set of all attributes \( \{ A_1, A_2, \ldots \} \) functionally determined by \( Z \) (that is, \( Z \rightarrow A_1 A_2 \ldots \))
- Algorithm for computing the closure
  - Start with closure = \( Z \)
  - If \( X \rightarrow Y \) is in \( F \) and \( X \) is already in the closure, then also add \( Y \) to the closure
  - Repeat until no more attributes can be added

A more complex example

\( \text{StudentGrade} \ (\text{SID}, \text{name}, \text{email}, \text{CID}, \text{grade}) \)

- \( \text{SID} \rightarrow \text{name}, \text{email} \)
- \( \text{email} \rightarrow \text{SID} \)
- \( \text{SID}, \text{CID} \rightarrow \text{grade} \)

(Not a good design, and we will see why later)

Example of computing closure

- \( F \) includes:
  - \( \text{SID} \rightarrow \text{name, email} \)
  - \( \text{email} \rightarrow \text{SID} \)
  - \( \text{SID, CID} \rightarrow \text{grade} \)
- \( \{ \text{CID, email} \}^+ = ? \)
- \( \text{email} \rightarrow \text{SID} \)
  - Add \( \text{SID} \); closure is now \( \{ \text{CID, email, SID} \} \)
- \( \text{SID} \rightarrow \text{name, email} \)
  - Add \( \text{name, email} \); closure is now \( \{ \text{CID, email, SID, name} \} \)
- \( \text{SID, CID} \rightarrow \text{grade} \)
  - Add \( \text{grade} \); closure is now all the attributes in \( \text{StudentGrade} \)

Using attribute closure

Given a relation \( R \) and set of FD’s \( F \)

- Does another FD \( X \rightarrow Y \) follow from \( F \)?
  - Compute \( X^+ \) with respect to \( F \)
  - If \( Y \subseteq X^+ \), then \( X \rightarrow Y \) follow from \( F \)
- Is \( K \) a key of \( R \)?
  - Compute \( K^+ \) with respect to \( F \)
  - If \( K^+ \) contains all the attributes of \( R \), \( K \) is a super key
  - Still need to verify that \( K \) is minimal (how?)

Rules of FD’s

- Armstrong’s axioms
  - Reflexivity: If \( Y \subseteq X \), then \( X \rightarrow Y \)
  - Augmentation: If \( X \rightarrow Y \), then \( Z \rightarrow YZ \) for any \( Z \)
  - Transitivity: If \( X \rightarrow Y \) and \( Y \rightarrow Z \), then \( X \rightarrow Z \)
- Rules derived from axioms
  - Splitting: If \( X \rightarrow YZ \), then \( X \rightarrow Y \) and \( X \rightarrow Z \)
  - Combining: If \( X \rightarrow Y \) and \( X \rightarrow Z \), then \( X \rightarrow YZ \)
Using rules of FD’s
Given a relation $R$ and set of FD’s $\mathcal{F}$

- Does another FD $X \rightarrow Y$ follow from $\mathcal{F}$?
  - Use the rules to come up with a proof

**Example:**
- $\mathcal{F}$ includes:
  - $SID \rightarrow name, email \rightarrow SID, SID, CID \rightarrow grade$
  - $email \rightarrow SID$ (given in $\mathcal{F}$)
  - $CID, email \rightarrow CID, S ID$ (augmentation)
  - $SID, CID \rightarrow grade$ (given in $\mathcal{F}$)
  - $CID, email \rightarrow grade$ (transitivity)

Non-key FD’s

- Consider a non-trivial FD $X \rightarrow Y$ where $X$ is not a super key
  - Since $X$ is not a super key, there are some attributes (say $Z$) that are not functionally determined by $X$

\[
\begin{align*}
X & \rightarrow Y \\
\{a, b, c\} & \rightarrow \{1, 2\}
\end{align*}
\]

That $a$ is always associated with $b$ is recorded by multiple rows: redundancy, update anomaly, deletion anomaly

Example of redundancy

- **StudentGrade** ($SID, name, email, CID, grade$)
- $SID \rightarrow name, email$

Unnecessary decomposition

- Fine: join returns the original relation
- Unnecessary: no redundancy is removed, and now $SID$ is stored twice!

Bad decomposition

- Association between $CID$ and $grade$ is lost
- Join returns more rows than the original relation
Lossless join decomposition

- Decompose relation $R$ into relations $S$ and $T$
  - $\text{attrs}(R) = \text{attrs}(S) \cup \text{attrs}(T)$
  - $S = \pi_{\text{attrs}(R)}(R)$
  - $T = \pi_{\text{attrs}(T)}(R)$
- The decomposition is a lossless join decomposition if, given known constraints such as FD’s, we can guarantee that $R = S \bowtie T$
- Any decomposition gives $R \subseteq S \bowtie T$ (why?)
  - A lossy decomposition is one with $R \subset S \bowtie T$

Questions about decomposition

- When to decompose
- How to come up with a correct decomposition (i.e., lossless join decomposition)

BCNF decomposition algorithm

- Find a BCNF violation
  - That is, a non-trivial FD $X \rightarrow Y$ in $R$ where $X$ is not a super key of $R$
- Decompose $R$ into $R_1$ and $R_2$, where
  - $R_1$ has attributes $X \cup Y$
  - $R_2$ has attributes $X \cup Z$, where $Z$ contains all attributes of $R$ that are in neither $X$ nor $Y$
- Repeat until all relations are in BCNF

Loss? But I got more rows!

- “Loss” refers not to the loss of tuples, but to the loss of information
  - Or, the ability to distinguish different original relations

An answer: BCNF

- A relation $R$ is in Boyce-Codd Normal Form if
  - For every non-trivial FD $X \rightarrow Y$ in $R$, $X$ is a super key
  - That is, all FDs follow from “key $\rightarrow$ other attributes”

BCNF decomposition example

- StudentGrade $(\text{SID, name, email, CID, grade})$
- Student $(\text{SID, name, email})$
- Grade $(\text{SID, CID, grade})$

BCNF
Another example

\[
\text{StudentGrade} (\text{SID}, \text{name}, \text{email}, \text{CID}, \text{grade})
\]

BCNF violation: \( \text{email} \rightarrow \text{SID} \)

\[
\text{StudentID} (\text{email}, \text{SID})
\]

BCNF

\[
\text{StudentGrade}' (\text{email}, \text{name}, \text{CID}, \text{grade})
\]

BCNF violation: \( \text{email} \rightarrow \text{name} \)

\[
\text{StudentName} (\text{email}, \text{name})
\]

BCNF

\[
\text{Grade} (\text{email}, \text{CID}, \text{grade})
\]

BCNF

Why is BCNF decomposition lossless

Given non-trivial \( X \rightarrow Y \) in \( R \) where \( X \) is not a super key of \( R \), need to prove:

- Anything we project always comes back in the join:
  \[ R \subseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R) \]
  - Sure; and it doesn’t depend on the FD
- Anything that comes back in the join must be in the original relation:
  \[ R \supseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R) \]
  - Proof makes use of the fact that \( X \rightarrow Y \)

Recap

- Functional dependencies: a generalization of the key concept
- Non-key functional dependencies: a source of redundancy
- BCNF decomposition: a method for removing redundancies
  - BCNF decomposition is a lossless join decomposition
- BCNF: schema in this normal form has no redundancy due to FD’s