Relational Database Design Theory
Part I

CPS 116
Introduction to Database Systems

Announcements (September 13)
- Homework #1 due this Thursday
- Course project assigned today
  - Choice of a “standard” or “open” course project
  - Two milestones (October 13 and November 10) and a final demo/report (December 6-13)

Motivation

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>CID</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Bart</td>
<td>CPS116</td>
</tr>
<tr>
<td>95</td>
<td>Lisa</td>
<td>CPS116</td>
</tr>
<tr>
<td>89</td>
<td>Lisa</td>
<td>CPS110</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- How do we tell if a design is bad, e.g., StudentEnroll (SID, name, 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

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>2</td>
</tr>
</tbody>
</table>

- Must be $b$
- Could be anything

FD examples

Address (street address, city, state, zip)
- street address, city, state $\rightarrow$ zip
- zip $\rightarrow$ city, state
- zip, state $\rightarrow$ zip?
  - This is a trivial FD
  - Trivial FD: LHS $\supseteq$ RHS
- zip $\rightarrow$ state, 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 \( \mathcal{F} \)
- Does another FD follow from \( \mathcal{F} \)?
  - Are some of the FD’s in \( \mathcal{F} \) 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 \( \mathcal{F} \) that hold in \( R \), and a set of attributes \( Z \) in \( R \):
  - The closure of \( Z \) (denoted \( Z^+ \)) with respect to \( \mathcal{F} \) 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 

A more complex example

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

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

Example of computing closure

- \( \mathcal{F} \) 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 \( \{ \text{SID, email, SID} \} \)
  - \( \text{SID} \rightarrow \text{name, email} \)
  - 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 \( \mathcal{F} \)
  - Does another FD \( X \rightarrow Y \) follow from \( \mathcal{F} \)?
    - Compute \( X^+ \) with respect to \( \mathcal{F} \)
    - If \( Y \subseteq X^+ \), then \( X \rightarrow Y \) follow from \( \mathcal{F} \)
  - Is \( K \) a key of \( R \)?
    - Compute \( K^+ \) with respect to \( \mathcal{F} \)
    - 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 \( XZ \rightarrow YZ \) 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 \( F \):
- Does another FD \( X \rightarrow Y \) follow from \( F \)?
  - Use the rules to come up with a proof
  - Example:
    - \( F \) includes:
      - \( SID \rightarrow name, email \rightarrow SID, SID, CID \rightarrow grade \)
      - \( CID, email \rightarrow grade \)
      - email \( \rightarrow \) SID (given in \( F \))
      - \( CID, email \rightarrow CID, SID \) (augmentation)
      - \( SID, CID \rightarrow grade \) (given in \( F \))
      - \( CID, email \rightarrow grade \) (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 \( X \)

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!

Decomposition
- Eliminates redundancy
- To get back to the original relation: \( \subseteq \)

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 $T$
  - $\text{attrs}(R) = \text{attrs}(S) \cup \text{attrs}(T)$
  - $S = \pi_{\text{attrs}(S)}(R)$
  - $T = \pi_{\text{attrs}(T)}(R)$
- 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

BCNF decomposition example

- $\text{StudentGrade}(\text{SID}, \text{name}, \text{email}, \text{CID}, \text{grade})$
  - BCNF violation: $\text{SID} \rightarrow \text{name}, \text{email}$
- $\text{Student}(\text{SID}, \text{name}, \text{email})$
  - BCNF
- $\text{Grade}(\text{SID}, \text{CID}, \text{grade})$
  - BCNF
Another example

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

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

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

StudentID is 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:

\(\checkmark\) Anything we project always comes back in the join:

\[R \subseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R)\]

\(\checkmark\) Sure; and it doesn’t depend on the FD

\(\checkmark\) Anything that comes back in the join must be in the original relation:

\[R \supseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R)\]

\(\checkmark\) Proof makes use of the fact that \(X \rightarrow Y\)

Recap

\(\checkmark\) Functional dependencies: a generalization of the key concept

\(\checkmark\) Non-key functional dependencies: a source of redundancy

\(\checkmark\) BCNF decomposition: a method for removing redundancies

\(\checkmark\) BCNF decomposition is a lossless join decomposition

\(\checkmark\) BCNF: schema in this normal form has no redundancy due to FD’s