Relational Database Design Theory
Part I

CPS 116
Introduction to Database Systems

Announcements

- Homework #1 due this Thursday (Sept. 9) at midnight
- Course project assigned today
  - First milestone due September 30
- Details of (optional) student presentations will be available this Thursday
- Let me know if you still do not have a Gradiance or DB2 account

Motivation

- 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>k</td>
<td>l</td>
</tr>
</tbody>
</table>

Must be $b$ Could be anything

FD examples

Address ($street\_address$, $city$, $state$, $zip$)

- Trivial FD: LHS $\supseteq$ RHS
- 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 functionally determined by $Z$
- Algorithm for computing the closure
  - Start with closure $= Z$
  - If $X \rightarrow Y$ is in $\mathcal{F}$ 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

$StudentGrade\ (SID, \ name, \ email, \ CID, \ grade)$

- $SID \rightarrow name, email$
- $email \rightarrow SID$
- $SID, CID \rightarrow grade$

- Not a good design, and we will see why later
Example of computing closure

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

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

\[
\begin{array}{|c|c|c|}
  \hline
  X & Y & Z \\
  \hline
  a & b & c_1 \\
  a & b & c_2 \\
  \vdots & \vdots & \vdots \\
  \hline
\end{array}
\]

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

Example of redundancy

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

\[
\begin{array}{|c|c|c|c|c|c|}
  \hline
  SID & name & email & CID & grade \\
  \hline
  124 & Bart & bart@fox.com & CPS114 & B- \\
  124 & Bart & bart@fox.com & CPS114 & B \\
  123 & Milhouse & milhouse@fox.com & CPS116 & B+ \\
  857 & Lisa & lisa@fox.com & CPS116 & A+ \\
  857 & Lisa & lisa@fox.com & CPS130 & A+ \\
  456 & Ralph & ralph@fox.com & CPS114 & C \\
  \vdots & \vdots & \vdots & \vdots & \vdots \\
  \hline
\end{array}
\]
Decomposition

- Eliminates redundancy
- To get back to the original relation:

```
SID name email CID grade
142 Bart bart@fox.com 142 CPS116 B-
123 Milhouse milhouse@fox.com 142 CPS114 B
356 Ralph ralph@fox.com 123 CPS116 A+
142 CPS116 A-
```

Unnecessary decomposition

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

```
SID name email
142 Bart bart@fox.com
123 Milhouse milhouse@fox.com
356 Ralph ralph@fox.com
```

Bad decomposition

```
SID CID grade
142 CPS116 B-
142 CPS114 B
123 CPS116 A+
142 CPS116 A-
356 CPS114 C
```
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 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 $R \subset S \bowtie T$

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

No way to tell which is the original relation

Questions about decomposition

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

- When to decompose
  - As long as some relation is not in BCNF
- How to come up with a correct decomposition
  - Always decompose on a BCNF violation
  - Then it is guaranteed to be a 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

$\text{StudentGrade} (\text{SID, name, email, CID, grade})$
$\text{BCNF violation: } \text{SID} \rightarrow \text{name, email}$

$\text{Student} (\text{SID, name, email})$
$\text{Grade} (\text{SID, CID, grade})$
Another example

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

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

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