Relational Database Design Theory

CPS 116
Introduction to Database Systems

Announcements (Tue. Sep. 9)

- Homework #1 due in one week
  - Need a help session this Friday or next Monday?
- Course project description available today
  - Choice of “standard” or “open”
  - One- to three-person team (approval needed beyond 3)
  - Two milestones + demo/report
  - Milestone #1 due in ~5 1/2 weeks, right after fall break

Motivation

<table>
<thead>
<tr>
<th>SID</th>
<th>name</th>
<th>CID</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>Bart</td>
<td>CPS116</td>
</tr>
<tr>
<td>142</td>
<td>Bart</td>
<td>CPS114</td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td>CPS116</td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td>CPS130</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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 $Y$.

$$
\begin{array}{ccc}
X & Y & Z \\
1 & 2 & 3 \\
2 & 3 & 4 \\
\end{array}
$$

Must be $b$

Could be anything

FD examples

- Address (street_address, city, state, zip)
- street_address, city, state → zip

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 $X \rightarrow Y$ 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

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

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

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

- $\{ \text{CID, email} \}^+ = ?$
- $\text{email} \rightarrow \text{SID}$
  - Add $\text{SID}$; closure is now $\{ \text{CID, email, SID} \}$
- $\text{SID} \rightarrow \text{name, email}$
  - Add $\text{name, email}$; closure is now $\{ \text{CID, email, SID, name} \}$
- $\text{SID}, \text{CID} \rightarrow \text{grade}$
  - Add $\text{grade}$; closure is now all the attributes in $\text{StudentGrades}$

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 $R$?

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 these rules, you can prove or disprove an FD given a set of FDs
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 $X$

\[
\begin{array}{ccc}
X & Y & Z \\
1 & 0 & 1 \\
2 & 1 & 2 \\
\vdots & \vdots & \vdots \\
\end{array}
\]

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

Example of redundancy

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

<table>
<thead>
<tr>
<th>SID</th>
<th>name</th>
<th>email</th>
<th>CID</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>Bart</td>
<td><a href="mailto:bart@fox.com">bart@fox.com</a></td>
<td>CPS116</td>
<td>B-</td>
</tr>
<tr>
<td>142</td>
<td>Bart</td>
<td><a href="mailto:bart@fox.com">bart@fox.com</a></td>
<td>CPS114</td>
<td>B</td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td><a href="mailto:lisa@fox.com">lisa@fox.com</a></td>
<td>CPS116</td>
<td>A+</td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td><a href="mailto:lisa@fox.com">lisa@fox.com</a></td>
<td>CPS114</td>
<td>A-</td>
</tr>
<tr>
<td>456</td>
<td>Ralph</td>
<td><a href="mailto:ralph@fox.com">ralph@fox.com</a></td>
<td>CPS114</td>
<td>C</td>
</tr>
</tbody>
</table>

- Eliminates redundancy

To get back to the original relation:

<table>
<thead>
<tr>
<th>SID</th>
<th>name</th>
<th>email</th>
<th>CID</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>Bart</td>
<td><a href="mailto:bart@fox.com">bart@fox.com</a></td>
<td>CPS116</td>
<td>B-</td>
</tr>
<tr>
<td>142</td>
<td>Bart</td>
<td><a href="mailto:bart@fox.com">bart@fox.com</a></td>
<td>CPS114</td>
<td>B</td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td><a href="mailto:lisa@fox.com">lisa@fox.com</a></td>
<td>CPS116</td>
<td>A+</td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td><a href="mailto:lisa@fox.com">lisa@fox.com</a></td>
<td>CPS114</td>
<td>A-</td>
</tr>
<tr>
<td>456</td>
<td>Ralph</td>
<td><a href="mailto:ralph@fox.com">ralph@fox.com</a></td>
<td>CPS114</td>
<td>C</td>
</tr>
</tbody>
</table>
Unnecessary decomposition

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

Bad decomposition

- Decompose relation R into relations S and T
  - attrs(R) = attrs(S) \ union \ attrs(T)
  - S = \pi_{attrs}(R)
  - T = \pi_{attrs}(R)
- The decomposition is a lossless join decomposition if, given known constraints such as FD’s, we can guarantee that R = S \bowtie T

Lossless join decomposition

- Decompose relation R into relations S and T
  - attrs(R) = attrs(S) \ union \ attrs(T)
  - S = \pi_{attrs}(R)
  - T = \pi_{attrs}(R)
- 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

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 → 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 (details next)
  - 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

```
StudentGrade (SID, name, email, CID, grade)
BCNF violation: SID $\rightarrow$ name, email

Student (SID, name, email)
BCNF

Grade (SID, CID, grade)
BCNF
```

Another example

```
StudentGrade (SID, name, email, CID, grade)
BCNF violation: email $\rightarrow$ SID

StudentID (email, SID)
BCNF

StudentGrade' (email, name, CID, grade)
BCNF violation: email $\rightarrow$ name

StudentName (email, name)
BCNF

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

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

BCNF = no redundancy?

- Student (\( SID, CID, \text{club} \))
  - Suppose your classes have nothing to do with the clubs you join
  - FD’s?
  - BCNF?
  - Redundancies?

<table>
<thead>
<tr>
<th>SID</th>
<th>CID</th>
<th>Club</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>CPS116</td>
<td>ballet</td>
</tr>
<tr>
<td>142</td>
<td>CPS114</td>
<td>sumo</td>
</tr>
<tr>
<td>142</td>
<td>CPS114</td>
<td>ballet</td>
</tr>
<tr>
<td>123</td>
<td>CPS116</td>
<td>chess</td>
</tr>
<tr>
<td>123</td>
<td>CPS116</td>
<td>golf</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Multivalued dependencies

- A multivalued dependency (MVD) 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 rows in \( R \) agree on all the attributes of \( X \), then we can swap their \( Y \) components and get two new rows that are also in \( R \).

MVD examples

Student (SID, CID, club)
- \( SID \rightarrow CID \)
- \( SID \rightarrow club \)
  - Intuition: given \( SID, CID \) and club are “independent”
- \( SID, CID \rightarrow club \)
  - Trivial: \( LHS \cup RHS = \) all attributes of \( R \)
- \( SID, CID \rightarrow SID \)
  - Trivial: \( LHS \supseteq RHS \)

Complete MVD + FD rules

- FD reflexivity, augmentation, and transitivity
- MVD complementation:
  - If \( X \rightarrow Y \), then \( X \rightarrow \text{attr}(R) - X - Y \)
- MVD augmentation:
  - If \( X \rightarrow Y \) and \( V \subseteq W \), then \( XW \rightarrow YV \)
- MVD transitivity:
  - If \( X \rightarrow Y \) and \( Y \rightarrow Z \), then \( X \rightarrow Z - Y \)
- Replication (FD is MVD):
  - If \( X \rightarrow Y \), then \( X \rightarrow Y \)
  - Try proving things using these!
- Coalescence:
  - If \( X \rightarrow Y \) and \( Z \subseteq Y \) and there is some \( W \) disjoint from \( Y \) such that \( W \rightarrow Z \), then \( X \rightarrow Z \)
An elegant solution: chase

- Given a set of FD’s and MVD’s D, does another dependency d (FD or MVD) follow from D?
- Procedure
  - Start with the hypothesis of d, and treat them as "seed" tuples in a relation
  - Apply the given dependencies in D repeatedly
    - If we apply an FD, we infer equality of two symbols
    - If we apply an MVD, we infer more tuples
  - If we infer the conclusion of d, we have a proof
  - Otherwise, if nothing more can be inferred, we have a counterexample

Proof by chase

- In R(A, B, C, D), does A → B and B → C imply that A → C?

<table>
<thead>
<tr>
<th>Have</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>a</td>
<td>b1</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A → B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B → C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>a</td>
</tr>
</tbody>
</table>

Another proof by chase

- In R(A, B, C, D), does A → B and B → C imply that A → C?

<table>
<thead>
<tr>
<th>Have</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>a</td>
<td>b1</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A → B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B → C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>a</td>
</tr>
</tbody>
</table>

In general, both new tuples and new equalities may be generated
4NF

A relation $R$ is in Fourth Normal Form (4NF) if
- For every non-trivial MVD $X \rightarrow Y$ in $R$, $X$ is a superkey
- That is, all FD’s and MVD’s follow from “key → other attributes” (i.e., no MVD’s and no FD’s besides key functional dependencies)

4NF is stronger than BCNF
- Because every FD is also a MVD

4NF decomposition algorithm

- Find a 4NF violation
  - A non-trivial MVD $X \rightarrow Y$ in $R$ where $X$ is not a superkey
- Decompose $R$ into $R_1$ and $R_2$, where
  - $R_1$ has attributes $X \cup Y$
  - $R_2$ has attributes $X \cup Z$ ($Z$ contains attributes not in $X$ or $Y$)
- Repeat until all relations are in 4NF

- Almost identical to BCNF decomposition algorithm
- Any decomposition on a 4NF violation is lossless
Summary

- Philosophy behind BCNF, 4NF: Data should depend on the key, the whole key, and nothing but the key!

- Other normal forms
  - 3NF: More relaxed than BCNF; will not remove redundancy if doing so makes FDs harder to enforce
  - 2NF: Slightly more relaxed than 3NF
  - 1NF: All column values must be atomic