Relational Database Design Theory

CompSci 316
Introduction to Database Systems

Announcements (Thu. Sep. 12)

- Homework #1 due next Tuesday
  - If you haven't activated Azure, do it now!
  - All-electronic submission
- Piazza is up—use it more
  - There is also a thread for forming project teams
- Location for Rishi’s office hours has changed

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
  - Update, insertion, deletion anomalies
- 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 $Y$.

**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, $Z \rightarrow A_1A_2 \ldots$)
  - 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

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

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

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

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

\[
\begin{array}{ccc}
X & Y & Z \\
a & b & c_1 \\
a & b & c_2 \\
& & ...
\end{array}
\]

That $b$ is always associated with $a$ is recorded by multiple rows: redundancy, update/insertion/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>CPS316</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>CPS310</td>
<td>A</td>
</tr>
<tr>
<td>123</td>
<td>Milhouse</td>
<td><a href="mailto:milhouse@fox.com">milhouse@fox.com</a></td>
<td>CPS316</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>CPS310</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>CPS330</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>CPS310</td>
<td>C</td>
</tr>
</tbody>
</table>

<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>CPS316</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>CPS310</td>
<td>A</td>
</tr>
<tr>
<td>123</td>
<td>Milhouse</td>
<td><a href="mailto:milhouse@fox.com">milhouse@fox.com</a></td>
<td>CPS316</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>CPS310</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>CPS330</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>CPS310</td>
<td>C</td>
</tr>
</tbody>
</table>


Decomposition

<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>CPS316</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>CPS310</td>
<td>A</td>
</tr>
<tr>
<td>123</td>
<td>Milhouse</td>
<td><a href="mailto:milhouse@fox.com">milhouse@fox.com</a></td>
<td>CPS316</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>CPS310</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>CPS330</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>CPS310</td>
<td>C</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>SID</th>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>Bart</td>
<td><a href="mailto:bart@fox.com">bart@fox.com</a></td>
</tr>
<tr>
<td>123</td>
<td>Milhouse</td>
<td><a href="mailto:milhouse@fox.com">milhouse@fox.com</a></td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td><a href="mailto:lisa@fox.com">lisa@fox.com</a></td>
</tr>
<tr>
<td>456</td>
<td>Ralph</td>
<td><a href="mailto:ralph@fox.com">ralph@fox.com</a></td>
</tr>
</tbody>
</table>

Bad decomposition

<table>
<thead>
<tr>
<th>SID</th>
<th>CID</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>CPS316</td>
<td>B-</td>
</tr>
<tr>
<td>142</td>
<td>CPS310</td>
<td>B</td>
</tr>
<tr>
<td>123</td>
<td>CPS316</td>
<td>B+</td>
</tr>
<tr>
<td>857</td>
<td>CPS330</td>
<td>A+</td>
</tr>
<tr>
<td>456</td>
<td>CPS310</td>
<td>C</td>
</tr>
</tbody>
</table>

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?)
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 (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 $(SID, name, email)$
- Grade $(SID, CID, grade)$
- BCNF

Another example

- StudentGrade $(SID, name, email, CID, grade)$
- BCNF violation: $email \rightarrow SID$
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 will make 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 ($SID$, $CID$, club)
  - Suppose your classes have nothing to do with the clubs you join
  - FD’s?
  - BCNF?
  - Redundancies?
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 $R$.

**Example:**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Must be in $R$ too

MVD examples

**Student (SID, CID, club)**

- $SID \rightarrow CID$

- $SID, CID \rightarrow club$
  - 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 $X \rightarrow \text{attrs}(R) - X - Y$
- 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 $\mathcal{D}$, does another dependency $d$ (FD or MVD) follow from $\mathcal{D}$?

▷ Procedure
  - Start with the hypothesis of $d$, and treat them as “seed” tuples in a relation
  - Apply the given dependencies in $\mathcal{D}$ 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 $A \rightarrow B$ and $B \rightarrow C$ imply that $A \rightarrow C$?

<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>$b_1$</td>
</tr>
<tr>
<td>$a$</td>
<td>$b_2$</td>
</tr>
</tbody>
</table>

$A \rightarrow B$
- $a$, $b_1$, $c_1$, $d_1$
- $a$, $b_2$, $c_2$, $d_2$

$B \rightarrow C$
- $a$, $b_2$, $c_1$, $d_2$
- $a$, $b_1$, $c_2$, $d_1$

$B \rightarrow C$
- $a$, $b_1$, $c_2$, $d_1$
- $a$, $b_1$, $c_1$, $d_2$

Another proof by chase

▷ In $R(A, B, C, D)$, does $A \rightarrow B$ and $B \rightarrow C$ imply that $A \rightarrow C$?

<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>$b_1$</td>
</tr>
</tbody>
</table>

$A \rightarrow B$
- $a$, $b_1$, $c_1$, $d_1$
- $a$, $b_2$, $c_2$, $d_2$

$B \rightarrow C$
- $b_1 = b_2$
- $c_1 = c_2$

In general, both new tuples and new equalities may be generated
Counterexample by chase

- In $R(A, B, C, D)$, does $A \rightarrow BC$ and $CD \rightarrow B$ imply that $A \rightarrow B$?

<table>
<thead>
<tr>
<th>Have</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b1 c1 d1</td>
<td>b1 = b2</td>
</tr>
<tr>
<td>a b2 c2 d2</td>
<td></td>
</tr>
</tbody>
</table>

$A \rightarrow BC$

Counterexample!

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 $R$ 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
### 4NF decomposition example

**Student (SID, CID, club)**

- **4NF violation**: $SID \rightarrow CID$

**Enroll (SID, CID)**

- **4NF**

**Join (SID, club)**

- **4NF**

### 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