Relational Database Design Theory

Introduction to Databases

CompSci 316 Spring 2017
Announcements (Wed. Feb. 1)

• Homework #1 due Monday 02/06 (11:59 pm)
Review: Motivation

- **redundancy** is bad
  - user name is recorded multiple times
- Leads to **update, insertion, deletion anomalies**
- Have a systematic approach to detecting and removing redundancy in designs
- **Dependencies, decompositions, and normal forms**
Review: Functional dependencies

• A functional dependency (FD) $X \rightarrow Y$
  • $X$ and $Y$ are sets of attributes in a relation $R$
  • 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>$c_1$</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$b$</td>
<td>$c_1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$Z$</th>
<th>$W$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
<td>$d_2$</td>
</tr>
<tr>
<td>$a$</td>
<td>$b_1$</td>
<td>$c$</td>
<td>$d_2$</td>
</tr>
</tbody>
</table>

$X \rightarrow Y$
$XY \rightarrow Z$

NOTE: You can only say which FDs do not hold in an instance
Cannot say which ones hold
FDs are given by schema : must be true for all instances (like keys)
Review: 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, $Z \rightarrow A_1A_2\ldots$

• $\{\text{gid, twitterid}\}^+$ = ?
  • twitterid $\rightarrow$ uid -------------- Closure grows to $\{\text{gid, twitterid, uid}\}$
  • uid $\rightarrow$ uname, twitterid -------------- Closure grows to $\{\text{gid, twitterid, uid, uname}\}$
  • uid, gid $\rightarrow$ fromDate -------------- Closure is now all attributes in UserJoinsGroup
Review: Superkeys and Keys

Given a relation $R$ and set of FD’s $\mathcal{F}$

- Compute $K^+$ with respect to $\mathcal{F}$

- If $K^+$ contains all the attributes of $R$, $K$ is a super key

- If $K$ is also minimal (no proper subset is a superkey), $K$ is a key
Review: Motivation of BCNF decomposition

• Non-key FDs cause redundancy

Here $X \rightarrow Y$

Detect such FDs where $X$ is not a superkey, and decompose into two relations

1. One relation gets $X, Y$ (X is a superkey there! this makes it lossless)
2. The other one gets $X, Z$ (in general $Z =$ everything else)

Note: you need to consider all FDs that can be inferred! not only the ones that are given
Review: BCNF decomposition example

UserJoinsGroup (uid, uname, twitterid, gid, fromDate)

BCNF violation: twitterid → uid

UserId (twitterid, uid)

BCNF

UserJoinsGroup' (twitterid, uname, gid, fromDate)

BCNF violation: twitterid → uname

twitterid, gid → fromDate

UserName (twitterid, uname)

BCNF

Member (twitterid, gid, fromDate)

BCNF

uid → uname, twitterid
twitterid → uid
uid, gid → fromDate

apply Armstrong’s axioms and rules!
Lossy and Lossless Decomposition

Check yourself!
if in one of the two new relations, the common join attributes is a superkey, then lossless
Review: Multi-valued Dependency motivation

- **User** 
  
  (uid, gid, place)

- No FD like uid → gid or uid → place

- Still redundancy

- Given a user, gid and place are independent

  e.g. given uid = 456, all combinations exist for

  \((abc, gov) \times (Springfield, Morocco)\)
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 the following:
  - Whenever two rows in \( R \) agree on all the attributes of \( X \),
  - then we can swap their \( Y \) components and get two rows that are also in \( R \).

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
<th>( Z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( b_1 )</td>
<td>( c_1 )</td>
</tr>
<tr>
<td>( a )</td>
<td>( b_2 )</td>
<td>( c_2 )</td>
</tr>
<tr>
<td>( a )</td>
<td>( b_2 )</td>
<td>( c_1 )</td>
</tr>
<tr>
<td>( a )</td>
<td>( b_1 )</td>
<td>( c_2 )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>
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 $X \rightarrow Y$
- 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 premise 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>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \rightarrow B$</td>
<td>$a$</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>$B \rightarrow C$</td>
<td>$a$</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>$d_1$</td>
</tr>
<tr>
<td></td>
<td>$a$</td>
<td>$b_2$</td>
<td>$c_1$</td>
<td>$d_1$</td>
</tr>
<tr>
<td></td>
<td>$a$</td>
<td>$b_2$</td>
<td>$c_1$</td>
<td>$d_1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Need:</th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \rightarrow C$</td>
<td>$a$</td>
<td>$b_1$</td>
<td>$c_2$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>$B \rightarrow C$</td>
<td>$a$</td>
<td>$b_2$</td>
<td>$c_1$</td>
<td>$d_2$</td>
</tr>
</tbody>
</table>
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: ( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( D )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( b_1 )</td>
<td>( c_1 )</td>
<td>( d_1 )</td>
</tr>
<tr>
<td>( a )</td>
<td>( b_2 )</td>
<td>( c_2 )</td>
<td>( d_2 )</td>
</tr>
</tbody>
</table>

| Need: \( c_1 = c_2 \) |

\( A \rightarrow B \quad b_1 = b_2 \)

\( B \rightarrow C \quad c_1 = c_2 \)

In general, with both MVD’s and FD’s, chase can generate both new tuples and new equalities.
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></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>$b_2$</td>
<td>$c_2$</td>
<td>$d_2$</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>$b_2$</td>
<td>$c_2$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>$d_2$</td>
</tr>
</tbody>
</table>

Have: $A \rightarrow BC$

Need: $b_1 = b_2$ ✗

Countercexample!

Note: the FD must hold on all instances, so showing one instance as a counterexample suffices!
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 $\rightarrow$ 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
- why? because trivially if two tuples have same $X$ value, they also have the same $Y$ value, no question in swapping the $Y$ values!
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 $X \cup Z$ (where $Z$ 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

User $(uid, gid, place)$

4NF violation: $uid \rightarrow gid$

Member $(uid, gid)$

4NF

<table>
<thead>
<tr>
<th>$uid$</th>
<th>$gid$</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>dps</td>
</tr>
<tr>
<td>456</td>
<td>abc</td>
</tr>
<tr>
<td>456</td>
<td>gov</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Visited $(uid, place)$

4NF

<table>
<thead>
<tr>
<th>$uid$</th>
<th>place</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>Springfield</td>
</tr>
<tr>
<td>142</td>
<td>Australia</td>
</tr>
<tr>
<td>456</td>
<td>Springfield</td>
</tr>
<tr>
<td>456</td>
<td>Morocco</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Summary

• Philosophy behind BCNF, 4NF: Data should depend on the key, the whole key, and nothing but the key!
  • You could have multiple keys though

• 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
Next: Project Mixer!