Relational Database Design Theory

CPS 116
Introduction to Database Systems

Announcements (Thu. Sep. 15)

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 $F$

- Does another FD follow from $F$?
- Are some of the FD’s in $F$ 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 $F$ that hold in $R$, and a set of attributes $Z$ in $R$:
  - The closure of $Z$ (denoted $Z^+$) with respect to $F$ 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 $F$ 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

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

(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 \( \text{SID} \); closure is now \( \{ \text{CID, email, SID} \} \)
- \( \text{SID} \rightarrow \text{name, email} \)
  - Add \( \text{name, email} \); closure is now \( \{ \text{CID, email, SID, name} \} \)
- \( \text{SID, CID} \rightarrow \text{grade} \)
  - Add \( \text{grade} \); closure is now all the attributes in \( \text{StudentGrade} \)

Using attribute closure

Given a relation \( R \) and set of FD’s \( F \)

- 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

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 $X$

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

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

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>123</td>
<td>Milhouse</td>
<td><a href="mailto:milhouse@fox.com">milhouse@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>CPS118</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>CPS110</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>

Decomposition

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

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

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 \( R \subset S \bowtie T \)
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 → 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

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

Another example

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

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 $X \bowtie Y$, where $X$ and $Y$ are sets of attributes in a relation $R$.
- $X \bowtie Y$ 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 $R$.

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

*Must be in $R$ too*

**MVD examples**

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

- $SID$, $CID \bowtie club$
  - Trivial: LHS $\cup$ RHS = all attributes of $R$
- $SID$, $CID \bowtie SID$
  - Trivial: LHS $\supset$ RHS

**Complete MVD + FD rules**

- FD reflexivity, augmentation, and transitivity
- MVD complementation: If $X \bowtie Y$, then $X \bowtie \text{attr}(R) - X - Y$
- MVD augmentation: If $X \bowtie Y$ and $V \subseteq W$, then $XW \bowtie YV$
- MVD transitivity: If $X \bowtie Y$ and $Y \bowtie Z$, then $X \bowtie Z - Y$
- Replication (FD is MVD): If $X \rightarrow Y$, then $X \bowtie Y$
  - Try proving things using these!
- Coalescence: If $X \bowtie Y$ 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 \( A \bowtie B \) and \( B \bowtie C \) imply that \( A \bowtie C \)?

<table>
<thead>
<tr>
<th>Have</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \bowtie B )</td>
<td></td>
</tr>
<tr>
<td>( B \bowtie C )</td>
<td></td>
</tr>
<tr>
<td>( B \bowtie C )</td>
<td></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</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \rightarrow B )</td>
<td>( c1 = c2 )</td>
</tr>
<tr>
<td>( B \rightarrow C )</td>
<td>( c1 = c2 )</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Have</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( b_1 )</td>
</tr>
<tr>
<td>( b )</td>
<td>( c_1 )</td>
</tr>
<tr>
<td>( c )</td>
<td>( d_1 )</td>
</tr>
<tr>
<td>( e )</td>
<td>( a )</td>
</tr>
<tr>
<td>( f )</td>
<td>( b_2 )</td>
</tr>
<tr>
<td>( g )</td>
<td>( c_2 )</td>
</tr>
<tr>
<td>( h )</td>
<td>( d_1 )</td>
</tr>
<tr>
<td>( i )</td>
<td>( e )</td>
</tr>
<tr>
<td>( j )</td>
<td>( f )</td>
</tr>
<tr>
<td>( k )</td>
<td>( g )</td>
</tr>
<tr>
<td>( l )</td>
<td>( h )</td>
</tr>
<tr>
<td>( m )</td>
<td>( i )</td>
</tr>
<tr>
<td>( n )</td>
<td>( j )</td>
</tr>
<tr>
<td>( o )</td>
<td>( k )</td>
</tr>
<tr>
<td>( p )</td>
<td>( l )</td>
</tr>
<tr>
<td>( q )</td>
<td>( m )</td>
</tr>
<tr>
<td>( r )</td>
<td>( n )</td>
</tr>
<tr>
<td>( s )</td>
<td>( o )</td>
</tr>
<tr>
<td>( t )</td>
<td>( p )</td>
</tr>
<tr>
<td>( u )</td>
<td>( q )</td>
</tr>
<tr>
<td>( v )</td>
<td>( r )</td>
</tr>
<tr>
<td>( w )</td>
<td>( s )</td>
</tr>
<tr>
<td>( x )</td>
<td>( t )</td>
</tr>
<tr>
<td>( y )</td>
<td>( u )</td>
</tr>
<tr>
<td>( z )</td>
<td>( v )</td>
</tr>
<tr>
<td>( \overline{a} )</td>
<td>( \overline{b}_1 )</td>
</tr>
<tr>
<td>( \overline{b} )</td>
<td>( \overline{c}_1 )</td>
</tr>
<tr>
<td>( \overline{c} )</td>
<td>( \overline{d}_1 )</td>
</tr>
<tr>
<td>( \overline{e} )</td>
<td>( \overline{a} )</td>
</tr>
<tr>
<td>( \overline{f} )</td>
<td>( \overline{b}_2 )</td>
</tr>
<tr>
<td>( \overline{g} )</td>
<td>( \overline{c}_2 )</td>
</tr>
<tr>
<td>( \overline{h} )</td>
<td>( \overline{d}_1 )</td>
</tr>
<tr>
<td>( \overline{i} )</td>
<td>( \overline{e} )</td>
</tr>
<tr>
<td>( \overline{j} )</td>
<td>( \overline{f} )</td>
</tr>
<tr>
<td>( \overline{k} )</td>
<td>( \overline{g} )</td>
</tr>
<tr>
<td>( \overline{l} )</td>
<td>( \overline{h} )</td>
</tr>
<tr>
<td>( \overline{m} )</td>
<td>( \overline{i} )</td>
</tr>
<tr>
<td>( \overline{n} )</td>
<td>( \overline{j} )</td>
</tr>
<tr>
<td>( \overline{o} )</td>
<td>( \overline{k} )</td>
</tr>
<tr>
<td>( \overline{p} )</td>
<td>( \overline{l} )</td>
</tr>
<tr>
<td>( \overline{q} )</td>
<td>( \overline{m} )</td>
</tr>
<tr>
<td>( \overline{r} )</td>
<td>( \overline{n} )</td>
</tr>
<tr>
<td>( \overline{s} )</td>
<td>( \overline{o} )</td>
</tr>
<tr>
<td>( \overline{t} )</td>
<td>( \overline{p} )</td>
</tr>
<tr>
<td>( \overline{u} )</td>
<td>( \overline{q} )</td>
</tr>
<tr>
<td>( \overline{v} )</td>
<td>( \overline{r} )</td>
</tr>
<tr>
<td>( \overline{w} )</td>
<td>( \overline{s} )</td>
</tr>
<tr>
<td>( \overline{x} )</td>
<td>( \overline{t} )</td>
</tr>
<tr>
<td>( \overline{y} )</td>
<td>( \overline{u} )</td>
</tr>
<tr>
<td>( \overline{z} )</td>
<td>( \overline{v} )</td>
</tr>
</tbody>
</table>

4NF

- A relation \( R \) is in Fourth Normal Form (4NF) if
  - For every non-trivial MVD \( X \supset Y \) 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 \( X \supset Y \) 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
4NF decomposition example

Student (SID, CID, club)

4NF violation: SID \(\not\subseteq\) CID

Enroll (SID, CID)

- SID 142 CPS116 ballet
- SID 142 CPS116 sumo
- SID 142 CPS114 ballet
- SID 142 CPS114 sumo
- SID 123 CPS116 chess
- SID 123 CPS116 golf
- ...

Join (SID, club)

- SID club ballet
- SID club sumo
- SID club chess
- SID club golf
- ...

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