Announcements

• Homework #1 out today
  – Due next Thursday in class
• Sign up to present a research paper
  – Sign-up sheet available in my office (D327) during my office hours
    • First-come, first-serve
  – Participation is voluntary
    • Allows you to drop your lowest homework grade
  – In groups of 2-4

Relational model: a review

• A database is a collection of relations (or tables)
• Each relation has a list of attributes (or columns)
• Each attribute has a domain (or type)
• Each relation contains a set of tuples (or rows)
Keys

- A set of attributes $K$ is a key for a relation $R$ if
  - In no instance of $R$ will two different tuples agree on all attributes of $K$
    - That is, $K$ is a “tuple identifier”
  - No proper subset of $K$ satisfies the above condition
    - That is, $K$ is minimal
- Example: $\text{Student} (\text{SID, name, age, GPA})$

More examples of keys

- $\text{Enroll (SID, CID)}$
  - $\text{Address (street_address, city, state, zip)}$

Schema versus data

<table>
<thead>
<tr>
<th>SID</th>
<th>name</th>
<th>age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>Bart</td>
<td>10</td>
<td>2.3</td>
</tr>
<tr>
<td>123</td>
<td>Milhouse</td>
<td>10</td>
<td>3.1</td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td>8</td>
<td>4.3</td>
</tr>
<tr>
<td>456</td>
<td>Ralph</td>
<td>8</td>
<td>2.3</td>
</tr>
</tbody>
</table>

- Is name a key of $\text{Student}$?
  - Yes?
  - No!
- Key declarations are part of the schema
Usage of keys

- More constraints on data, fewer mistakes
- Look up a row by its key value
  - Many selection conditions are “key = value”
- “Pointers”
  - Example: Enroll (SID, CID)

- Many join conditions are “key = key value stored in another table”

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 of \(X\), they must also agree on all attributes of \(Y\)

FD examples

Address (street_address, city, state, zip)
Keys redefined using FDs

A set of attributes $K$ is a key for a relation $R$ if

- That is, $K$ is a "super key"
- No proper subset of $K$ satisfies the above condition
  - That is, $K$ is minimal

Reasoning with FDs

Given a relation $R$ and set of FDs $F$

- Does another FD follow from $F$?
  - Are some of the FDs in $F$ redundant (because 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 FDs $F$ that holds in $R$, and a set of attributes $Z$ in $R$: The closure of $Z$ with respect to $F$ (denoted $Z^*$) is the set of all attributes functionally determined by $Z$
- Algorithm for computing the closure
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

- $\{ CID, email \}^* = ?$

Using attribute closure

Given a relation $R$ and set of FDs $F$

- Does another FD $X \rightarrow Y$ follow from $F$?

- Is $K$ a key of $R$?
Rules of FDs

- 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 FDs

Given a relation \( R \) and set of FDs \( F \)

- Does another FD \( X \rightarrow Y \) follow from \( F \) ?
  - Use the rules to come up with a proof
  - Example: \( CID, email \rightarrow grade? \)

Non-key FDs

- 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 \( X \)

<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</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>

The fact that “a” is always associated with “b” is recorded in multiple rows: redundancy!
Problems with 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>CPS 216</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>CPS 214</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>CPS 216</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>CPS 216</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>CPS 130</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>CPS 214</td>
<td>C</td>
</tr>
</tbody>
</table>

Decomposition

\[ SID \rightarrow name, email \]

\[ SID \rightarrow CID, grade \]

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

Unnecessary 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></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>Milhouse</td>
<td><a href="mailto:milhouse@fox.com">milhouse@fox.com</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td><a href="mailto:lisa@fox.com">lisa@fox.com</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>456</td>
<td>Ralph</td>
<td><a href="mailto:ralph@fox.com">ralph@fox.com</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lossless join decomposition

- Suppose that $R$ is decomposed into $S$ and $T$
  - $\text{attrs}(R) = \text{attrs}(S) \cup \text{attrs}(T)$
  - $S = \pi_{\text{attrs}(S)}(R)$
  - $T = \pi_{\text{attrs}(T)}(R)$
- It is a lossless join decomposition if, given constraints such as FDs, we can guarantee $R = S \bowtie T$

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

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’s 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 $X \cup Z$ ($Z$ contains all attributes of $R$ that are in neither $X$ nor $Y$)
- Repeat until all relations are in BCNF
BCNF decomposition example

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

Another example

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

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:
    \[ R \subseteq \Pi_{XY}(R) 
    \rightarrow \Pi_{XZ}(R) \]
    - Sure; and it doesn’t depend on the FD
  - Anything that comes back in the join must be in the original relation:
    \[ R \supseteq \Pi_{XY}(R) 
    \rightarrow \Pi_{XZ}(R) \]
Yet another example

- Address (street_address, city, state, zip)
  - street_address, city, state → zip
  - zip → city, state
- Keys
- BCNF?

To decompose, or not to decompose

Address₁ (zip, city, state)
Address₂ (street_address, zip)
- FDs in Address₁
- FDs in Address₂

“Elegant” solution

- Define the problem away!
- R is in Third Normal Form (3NF) if for every non-trivial FD X → A, either
  - X is super key of R, or
  - A is a member of at least one key of R
- So Address is already in 3NF
- Tradeoff
Recap

- Identifying tuples: keys
- Generalizing the key concept: FDs
- Non-key FDs: redundancy
- Avoiding redundancy: BCNF decomposition
- Preserving FDs: 3NF

What’s next

- Another kind of dependency and normal form
- A comprehensive design example
- SQL basics