Introduction to Data Management

Lecture #7
(Relational Design Theory)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

- Thank you, Shiva, for covering last week!
- HW#2 is underway...! (From HW #1 solution)
  - Use your best judgement on data types and NULLs
  - Capture all the capturable ICs (PK/FK/UNIQUE)
- A few logistical notes (office hours, discussions)
  - ~450 = Hybrid F2F/online (Piazza) class (☺)
  - Definitely bring your questions to discussion section meetings (including the just-finished HW solution)
- Today’s plan:
  - Relational DB design theory
  - Disclaimer: Not the most exciting part of CS122A… ☺
So, Given a Relational Schema...

- How do I know if my relational schema is a “good” logical database design or not?
  - What might make it “not good”?
  - How can I fix it, if indeed it’s “not good”?
  - How “good” is it, after I’ve fixed it?

- Note that your relational schema might have come from one of several places
  - You started from an E-R model (but maybe that model was “wrong” or incomplete in some way?)
  - You went straight to relational in the first place
  - It’s not your schema – you inherited it! 😊

Ex: Wisconsin Sailing Club

Proposed bad schema design:

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
<th>date</th>
<th>bid</th>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45.0</td>
<td>10/10/98</td>
<td>101</td>
<td>Interlake</td>
<td>blue</td>
</tr>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45.0</td>
<td>10/10/98</td>
<td>102</td>
<td>Interlake</td>
<td>red</td>
</tr>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45.0</td>
<td>10/8/98</td>
<td>103</td>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45.0</td>
<td>10/7/98</td>
<td>104</td>
<td>Marine</td>
<td>red</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
<td>11/10/98</td>
<td>102</td>
<td>Interlake</td>
<td>red</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
<td>11/6/98</td>
<td>103</td>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
<td>11/12/98</td>
<td>104</td>
<td>Marine</td>
<td>red</td>
</tr>
</tbody>
</table>

**Review:** Bad design due to redundancy and its problems.
**Ex: Wisconsin Sailing Club**

Proposed good schema design:

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>10/10/98</td>
</tr>
<tr>
<td>22</td>
<td>102</td>
<td>10/10/98</td>
</tr>
<tr>
<td>22</td>
<td>103</td>
<td>10/8/98</td>
</tr>
<tr>
<td>22</td>
<td>104</td>
<td>10/7/98</td>
</tr>
<tr>
<td>31</td>
<td>102</td>
<td>11/10/98</td>
</tr>
<tr>
<td>31</td>
<td>103</td>
<td>11/6/98</td>
</tr>
<tr>
<td>31</td>
<td>104</td>
<td>11/12/98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bid</th>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Interlake</td>
<td>blue</td>
</tr>
<tr>
<td>102</td>
<td>Interlake</td>
<td>red</td>
</tr>
<tr>
<td>103</td>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>104</td>
<td>Marine</td>
<td>red</td>
</tr>
</tbody>
</table>

**Review:** Good design due to elimination of redundancy.

---

**The Evils of Redundancy (or: The Evils of Redundancy)**

- **Redundancy** is at the root of several problems associated with relational schemas:
  - Redundant storage (space)
  - Insert/delete/update anomalies
  
  *A good rule to follow:* **“One fact, one place!”**

- **Functional dependencies** can help in identifying problem schemas and suggesting refinements.

- Main refinement technique: *decomposition*, e.g., replace R(ABCD) with R1(AB) + R2(BCD)

- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - Does the decomposition cause any problems?
**Functional Dependencies (FDs)**

- A **functional dependency** $X \rightarrow Y$ holds over relation $R$ if, for every allowable instance $r$ of $R$:
  - For $t_1$ and $t_2$ in $r$, $t_1.X = t_2.X$ implies $t_1.Y = t_2.Y$
  - I.e., given two tuples in $r$, if the $X$ values agree, then their $Y$ values must also agree. ($X$ and $Y$ can be sets of attributes.)

- An FD is a statement about **all** allowable relations.
  - Identified based on *application semantics* (similar to E-R).
  - Given some instance $r_1$ of $R$, we can check to see if it violates some FD $f$, but we cannot tell if $f$ holds over $R$!

- Saying $K$ is a candidate key for $R$ means that $K \rightarrow R$
  - Note: $K \rightarrow R$ alone does not require $K$ to be **minimal**! If $K$ is minimal, then $K$ is a candidate key (else it’s a super key).

---

**Example: Constraints on an Entity Set**

- Suppose you’re given a relation called **HourlyEmps**:
  - **HourlyEmps** $(ssn, name, lot, rating, hrly_wages, hrs_worked)$
- **Notation**: Let’s denote this relation schema by simply listing the attributes: **SNLRWH**
  - This is really the **set** of attributes {$S,N,L,R,W,H$}.
  - Sometimes, we will refer to **all** attributes of a relation by using the relation name (e.g., HourlyEmps for SNLRWH).

- Suppose we also have some FDs on HourlyEmps:
  - $ssn$ is the key: **$S \rightarrow SNLRWH$**
  - $rating$ determines $hrly_wages$: **$R \rightarrow W$**
Example (Cont’d.)

- Problems due to $R \rightarrow W$
  - **Update anomaly:** What if we change $W$ in just the 1st tuple of SNLRWH?
  - **Insertion anomaly:** What if we want to insert a new employee and don’t know the proper hourly wage for his or her rating?
  - **Deletion anomaly:** If we delete all employees with rating 5, we lose the stored information about the wage for rating 5!

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

HourlyEmps2

Wages

<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

How about two smaller tables?

Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:
  - $ssn \rightarrow did, did \rightarrow lot \implies ssn \rightarrow lot$
  - (Translation: Matching $ssns$ imply matching $lots$.)

- An FD $f$ is **implied by** a set of FDs $F$ if $f$ holds whenever all FDs in $F$ hold.
  - $F^+ = \text{closure of } F$ is the set of all FDs that are implied by $F$.

- **Armstrong’s Axioms** ($X$, $Y$, $Z$ are sets of attributes):
  - **Reflexivity:** If $X \subseteq Y$, then $Y \rightarrow X$
  - **Augmentation:** If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any $Z$
  - **Transitivity:** If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

- **These are sound and complete** inference rules for FDs!
Armstrong’s Axioms: Examples

<table>
<thead>
<tr>
<th>pno</th>
<th>name</th>
<th>title</th>
<th>state</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sandy</td>
<td>Professor</td>
<td>CA</td>
<td>92697</td>
</tr>
<tr>
<td>2</td>
<td>Joe</td>
<td>Jim Gray Professor</td>
<td>CA</td>
<td>94720</td>
</tr>
<tr>
<td>3</td>
<td>Anhai</td>
<td>Professor</td>
<td>WI</td>
<td>53706</td>
</tr>
<tr>
<td>4</td>
<td>Alex</td>
<td>Associate Professor</td>
<td>CA</td>
<td>92697</td>
</tr>
</tbody>
</table>

- **Reflexivity**: If \( X \subseteq Y \) then \( Y \rightarrow X \):
  - \( \text{zip} \subseteq (\text{zip}, \text{name}) \), so \( (\text{zip}, \text{name}) \rightarrow \text{zip} \).

- **Augmentation**: If \( X \rightarrow Y \) then \( XZ \rightarrow YZ \) for any \( Z \):
  - \( \text{zip} \rightarrow \text{state} \), so \( (\text{zip}, \text{title}) \rightarrow (\text{state}, \text{title}) \).

- **Transitivity**: If \( X \rightarrow Y \) and \( Y \rightarrow Z \) then \( X \rightarrow Z \):
  - \( \text{pno} \rightarrow \text{zip} \) and \( \text{zip} \rightarrow \text{state} \), so \( \text{pno} \rightarrow \text{state} \).

**Reasoning About FDs (Cont’d.)**

(Recall: “two matching X’s always have the same Y”)

- A few additional rules (which follow from AA):
  - **Union**: If \( X \rightarrow Y \) and \( X \rightarrow Z \), then \( X \rightarrow YZ \)
  - **Decomposition**: If \( X \rightarrow YZ \), then \( X \rightarrow Y \) and \( X \rightarrow Z \)

- Example: \( \text{Contracts} (cid, sid, pjid, did, pid, qty, value) \), and:
  - The contract id is the key: \( C \rightarrow \text{CSJDPQV} \)
  - A project purchases each part using single contract: \( \text{JP} \rightarrow C \)
  - A dept purchases at most one part from a supplier: \( \text{SD} \rightarrow P \)

- \( \text{JP} \rightarrow C, C \rightarrow \text{CSJDPQV} \) imply \( C \rightarrow \text{CSJDPQV} \)
- \( \text{SD} \rightarrow P \) implies \( \text{SDJ} \rightarrow \text{JP} \) (New candidate keys...!)
- \( \text{SDJ} \rightarrow \text{JP}, \text{JP} \rightarrow \text{CSJDPQV} \) imply \( \text{SD} \rightarrow \text{CSJDPQV} \)
**Reasoning About FDs (Examples)**

Let’s consider $R(ABCDE)$, $F = \{A \rightarrow B, B \rightarrow C, C D \rightarrow E\}$

- Let’s work our way towards inferring $F^+$ ...

1. $A \rightarrow B$ (given)
2. $B \rightarrow C$ (given)
3. $C D \rightarrow E$ (given)
4. $A \rightarrow C$ (a, b, and transitivity)
5. $B D \rightarrow C D$ (b and augmentation)
6. $B D \rightarrow E$ (b and augmentation)
7. $A D \rightarrow C D$ (d and augmentation)
8. $A D \rightarrow C$ (g and decomposition)
9. $A D \rightarrow D$ (g and decomposition)
10. $A D \rightarrow B$ (k and decomposition)
11. $A D \rightarrow A$ (a and reflexivity)
12. $A D \rightarrow ABCDE$ (h, i, j, l, m, and union)

**Candidate key!**

**Note:** If some attribute $X$ is not on the RHS of any initial FD, then $X$ must be part of the key!