Relational Database Design Theory (I)

Introduction to Database Management

CS348 Fall 2022
Announcements (Tue. Oct 18)

• Assignment 1’s grade was released last Thur
  • Partial solution is available on Learn
  • Appeal deadline is this Thur

• Milestone 1 is due this Thur, Oct 20, 11:59pm
  • Basic score is 45 points, capped by 49 points
  • Contribute $\frac{\min(s1,49)}{45} \times 30$ to the final project score

• Assignment 2 is released
  • Cover lectures till lecture 10
  • Due by Thur, Oct 27, 11:59pm
Design process – where are we?

• Schema refinement

Conceptual Design ➔ Conceptual Schema (E/R model) ➔ Logical Design ➔ Logical Schema (Relational model)

• What are relational design principles?
A Parts/Suppliers database example

- Each type of part has a name and an identifying number and may be supplied by zero or more suppliers.
- Each supplier has an identifying number, a name, and a contact location for ordering parts.
- Each supplier may offer the part at a different price.
Parts/Suppliers example (cont.)

• An instance

<table>
<thead>
<tr>
<th>Suppliers</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sno</td>
<td>Sname</td>
<td>City</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
</tr>
<tr>
<td>S2</td>
<td>Budd</td>
<td>Hull</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supplies</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sno</td>
<td>Pno</td>
</tr>
<tr>
<td>S1</td>
<td>P1</td>
</tr>
<tr>
<td>S1</td>
<td>P2</td>
</tr>
<tr>
<td>S1</td>
<td>P3</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
</tr>
</tbody>
</table>

Parts

<table>
<thead>
<tr>
<th>Pno</th>
<th>Pname</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Bolt</td>
</tr>
<tr>
<td>P2</td>
<td>Nut</td>
</tr>
<tr>
<td>P3</td>
<td>Screw</td>
</tr>
</tbody>
</table>
Alternate Parts/Suppliers database

Supplied_Items

<table>
<thead>
<tr>
<th>Sno</th>
<th>Sname</th>
<th>City</th>
<th>Pno</th>
<th>Pname</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P1</td>
<td>Bolt</td>
<td>0.50</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P2</td>
<td>Nut</td>
<td>0.25</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P3</td>
<td>Screw</td>
<td>0.30</td>
</tr>
<tr>
<td>S2</td>
<td>Budd</td>
<td>Hull</td>
<td>P3</td>
<td>Screw</td>
<td>0.40</td>
</tr>
</tbody>
</table>
Change anomalies

• The single-table schema suffers from:
  • **Update anomalies** (e.g. change supplier name)
  • **Insert anomalies** (e.g. add a new item)
  • **delete anomalies** (e.g. S1 no longer supplies Nut)
  • Likely increase in space requirements

<table>
<thead>
<tr>
<th>Sno</th>
<th>Sname</th>
<th>City</th>
<th>Pno</th>
<th>Pname</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P1</td>
<td>Bolt</td>
<td>0.50</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P2</td>
<td>Nut</td>
<td>0.25</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td>P3</td>
<td>Screw</td>
<td>0.30</td>
</tr>
<tr>
<td>S2</td>
<td>Budd</td>
<td>Hull</td>
<td>P3</td>
<td>Screw</td>
<td>0.40</td>
</tr>
</tbody>
</table>
Change anomalies

• The single-table schema suffers from:
  • Update anomalies (e.g. change supplier name)
  • Insert anomalies (e.g. add a new item)
  • delete anomalies (e.g. S1 no longer supplies Nut)
  • Likely increase in space requirements

• The multi-table schema does not have these problems.
Another alternate

• Is more tables always better?

<table>
<thead>
<tr>
<th>Snos</th>
<th>Sname</th>
<th>Cities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sno</td>
<td>Sname</td>
<td>City</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
</tr>
<tr>
<td>S2</td>
<td>Budd</td>
<td>Hull</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pnums</th>
<th>Pnames</th>
<th>Prices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pnum</td>
<td>Pname</td>
<td>Price</td>
</tr>
<tr>
<td>I1</td>
<td>Bolt</td>
<td>0.50</td>
</tr>
<tr>
<td>I2</td>
<td>Nut</td>
<td>0.25</td>
</tr>
<tr>
<td>I3</td>
<td>Screw</td>
<td>0.30</td>
</tr>
</tbody>
</table>

• Information about relationships is lost
Designing good databases

• Goals
  • A methodology for evaluating schemas (detecting anomalies)
  • A methodology for transforming bad schemas into good ones

• How do we know an anomaly exists?
• What should we do if an anomaly exists?

<table>
<thead>
<tr>
<th>Supplied_Items</th>
<th>Sno</th>
<th>Sname</th>
<th>City</th>
<th>Pno</th>
<th>Pname</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td></td>
<td>P1</td>
<td>Bolt</td>
<td>0.50</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td></td>
<td>P2</td>
<td>Nut</td>
<td>0.25</td>
</tr>
<tr>
<td>S1</td>
<td>Magna</td>
<td>Ajax</td>
<td></td>
<td>P3</td>
<td>Screw</td>
<td>0.30</td>
</tr>
<tr>
<td>S2</td>
<td>Budd</td>
<td>Hull</td>
<td></td>
<td>P3</td>
<td>Screw</td>
<td>0.40</td>
</tr>
</tbody>
</table>

Schema decomposition: avoid anomalies while retaining all info in the instances.

Integrity constraints (e.g. dependencies between attributes) → lead to anomalies
Design Theory

• Detect anomalies: Functional dependencies

• Repair anomalies: Schema decomposition
Functional dependencies

- Consider the following relation schema

<table>
<thead>
<tr>
<th>SIN</th>
<th>PNum</th>
<th>Hours</th>
<th>EName</th>
<th>PName</th>
<th>PLoc</th>
<th>Allowance</th>
</tr>
</thead>
</table>

1. SIN determines employee name
2. Project number determines project name and location
3. Allowances are always the same for the same number of hours at the same location

- A functional dependency (FD) has the form $X \rightarrow Y$, where $X$ and $Y$ are sets of attributes in a relation $R$
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 $Y$.

- If $X$ is a superkey of $R$, then $X \rightarrow R$ (all the attributes).

```
<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>?</td>
</tr>
</tbody>
</table>
```

Must be $b$ Could be anything
Functional dependencies

• Consider the following relation schema

<table>
<thead>
<tr>
<th>SIN</th>
<th>PNum</th>
<th>Hours</th>
<th>EName</th>
<th>PName</th>
<th>PLoc</th>
<th>Allowance</th>
</tr>
</thead>
</table>

1. SIN determines employee name
2. Project number determines project name and location
3. Allowances are always the same for the same number of hours at the same location

• How about SIN and EName determines Ename?
  • Trivial FD

\[ \text{SIN} \rightarrow \text{EName} \]
\[ \text{PNum} \rightarrow \text{PName}, \text{PLoc} \]
\[ \text{PLoc, Hours} \rightarrow \text{Allowance} \]

\[ \text{SIN, EName} \rightarrow \text{EName} \]
Closure of FD sets

• How do we know what additional FDs hold in a schema?

• A set of FDs \( \mathcal{F} \) logically implies a FD \( X \rightarrow Y \) if \( X \rightarrow Y \) holds in all instances of \( R \) that satisfy \( \mathcal{F} \).

• The closure of a FD set \( \mathcal{F} \) (denoted \( \mathcal{F}^+ \)):
  • The set of all FDs that are logically implied by \( \mathcal{F} \)
  • Informally, \( \mathcal{F}^+ \) includes all of the FDs in \( \mathcal{F} \), i.e., \( \mathcal{F} \subseteq \mathcal{F}^+ \), plus any dependencies they imply.
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
  • Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
  • Union: 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
Example for proving a FD

Prove $\text{SIN}, \text{PNum} \rightarrow \text{Allowance}$

1. $\text{SIN}, \text{PNum} \rightarrow \text{Hours} \ (\in \mathcal{F})$
2. $\text{PNum} \rightarrow \text{PName, PLoc} \ (\in \mathcal{F})$
3. $\text{PLoc, Hours} \rightarrow \text{Allowance} \ (\in \mathcal{F})$

$\mathcal{F}$ includes:
- $\text{SIN, PNum} \rightarrow \text{Hours}$
- $\text{SIN} \rightarrow \text{EName}$
- $\text{PNum} \rightarrow \text{PName, PLoc}$
- $\text{PLoc, Hours} \rightarrow \text{Allowance}$
Example for proving a FD

Prove $\text{SIN, PNum} \rightarrow \text{Allowance}$

1. $\text{SIN, PNum} \rightarrow \text{Hours} \ (\in \mathcal{F})$
2. $\text{PNum} \rightarrow \text{PName, PLoc} \ (\in \mathcal{F})$
3. $\text{PLoc, Hours} \rightarrow \text{Allowance} \ (\in \mathcal{F})$
4. $\text{SIN, PNum} \rightarrow \text{PNum} \ (\text{reflexivity})$
5. $\text{SIN, PNum} \rightarrow \text{PName, PLoc} \ (\text{transitivity, 4 and 2})$
6. $\text{SIN, PNum} \rightarrow \text{PLoc} \ (\text{decomposition, 5})$
7. $\text{SIN, PNum} \rightarrow \text{PLoc, Hours} \ (\text{union, 6 and 1})$
8. $\text{SIN, PNum} \rightarrow \text{Allowance} \ (\text{transitivity, 7 and 3})$

$\mathcal{F}$ includes:

- $\text{SIN, PNum} \rightarrow \text{Hours}$
- $\text{SIN} \rightarrow \text{EName}$
- $\text{PNum} \rightarrow \text{PName, PLoc}$
- $\text{PLoc, Hours} \rightarrow \text{Allowance}$
Example for proving a FD

Prove $SIN, PNum \rightarrow Allowance$

1. $SIN, PNum \rightarrow Hours (\in \mathcal{F})$
2. $PNum \rightarrow PName, PLoc (\in \mathcal{F})$
3. $PLoc, Hours \rightarrow Allowance (\in \mathcal{F})$
4. $SIN, PNum \rightarrow PNum$ (reflexivity)
5. $SIN, PNum \rightarrow PName, PLoc$ (transitivity, 4 and 2)
6. $SIN, PNum \rightarrow PLoc$ (decomposition, 5)
7. $SIN, PNum \rightarrow PLoc, Hours$ (union, 6 and 1)
8. $SIN, PNum \rightarrow Allowance$ (transitivity, 7 and 3)

$\mathcal{F}$ includes:
- $SIN, PNum \rightarrow Hours$
- $SIN \rightarrow EName$
- $PNum \rightarrow PName, PLoc$
- $PLoc, Hours \rightarrow Allowance$

Attribute closure of $\{SIN, PNum\}$
Attribute closure

• The closure of attributes $Z$ in a relation $R$ (denoted $Z^+$) with respect to a set of FDs, $\mathcal{F}$, 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

Compute$Z^+(Z, \mathcal{F})$:

• 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 new attributes can be added
Example for computing attribute closure

**Compute** $Z^+ ({PNum, Hours}, F)$:

$F$ includes:
- $SIN, PNum \rightarrow Hours$
- $SIN \rightarrow EName$
- $PNum \rightarrow PName, PLoc$
- $PLoc, Hours \rightarrow Allowance$

<table>
<thead>
<tr>
<th>FD</th>
<th>$Z^+$</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial</td>
<td>$PNum, Hours$</td>
</tr>
<tr>
<td>$PNum \rightarrow PName, PLoc$</td>
<td>$PNum, Hours, PName, PLoc$</td>
</tr>
<tr>
<td>$PLoc, Hours \rightarrow Allowance$</td>
<td>$PNum, Hours, PName, PLoc, Allowance$</td>
</tr>
</tbody>
</table>

$PNum, Hours \rightarrow PLoc, Allowance$
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 $X \rightarrow Y$ follows 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?) [Exercise]
    - Hint: check the attribute closure of its proper subset.
Design Theory

• Detect anomalies: Functional dependencies
  • Closure of FDs (rules, e.g. Armstrong’s axioms)
  • Attribute closure

• Repair anomalies: Schema decomposition
  • (next lecture)