CS 133: Databases

Fall 2016
Lec 21 – 11/21
Prof. Beth Trushkowsky

Administrivia

• No office hours this Wednesday
  – By appointment on Tuesday

• Current problem set due Thursday after break

• Lab 4 due this Wednesday night
• Lab 5 out after break

Goals for Today

• Understand the issues that can occur from a poorly designed relational schema

• Learn about the goals and process of schema refinement

• Discuss the role of functional dependencies in discovering and fixing issues in a design

Review: Database Design

• Requirements Analysis
  – user needs; what must database do?

• Conceptual Design
  – high level description (often done w/ER model)

• Logical Design
  – translate ER into DBMS data model

• Schema Refinement
  – consistency, normalization

• Physical Design - indexes, disk layout

• Security Design - who accesses what
Integrity constraints: challenges

CREATE TABLE Owns
(ssn CHAR(11),
  bar_name CHAR(20),
  PRIMARY KEY (bar_name),
  FOREIGN KEY (ssn)
    REFERENCES Drinkers,
  FOREIGN KEY (bar_name)
    REFERENCES Bars(name))

Is it possible to capture this participation constraint in Owns?

Implications of adding NOT NULL to ssn?

What if we wanted Drinkers to have total participation in Owns?

What about enforcing Drinkers are at least 21 years old?!
Triggers

• Trigger: procedure that starts automatically if specified changes occur to the DBMS
• Three parts:
  – Event (activates the trigger)
  – Condition (tests whether the triggers should run)
  – Action (what happens if the trigger runs)

• A database with a collection of triggers often called an active database

Triggers: Example (SQL:1999)

CREATE TRIGGER youngSailorUpdate
AFTER INSERT ON SAILORS
REFERENCING NEW TABLE NewSailors
FOR EACH STATEMENT
  INSERT
  INTO YoungSailors(sid, name, age, rating)
  SELECT sid, name, age, rating
  FROM NewSailors N
  WHERE N.age <= 18

Other DDL Statements

• Alter Table
  – use to add/remove columns, constraints, rename things ...
• Drop Table
  – Compare to “Delete * From Table”
• Create/Drop View
• Create/Drop Index
• Grant/Revoke privileges
  – SQL has an authorization model for saying who can read/modify/delete etc. data and who can grant and revoke privileges!

Combos: Entities and Relationships

• For one-to-many relationship, combining entity set and relationship set into one relation helped us capture participation constraint

• What about combining Bars and Sells?

Bar_Sells (name, beer_name, address, price)

Redundancy!
Schema Refinement

• Start with initial relational schema, either from scratch or from E/R modeling

• Schema refinement: could there be data redundancy issues?

• Next: why is redundancy bad

Example: Hourly_Emps

• Consider a relation obtained from Hourly_Emps:
  Hourly_Emps (ssn, name, lot, rating, wage_per_hr, hrs_per_wk)

Can denote a relation schema by listing its attributes, e.g., SNLRWH
This is really the set of attributes \{S, N, L, R, W, H\}

• Things we know (from semantics)
  – ssn uniquely identifies an employee (is a key)
  – An employee’s rating determines his/her wage_per_hr

Redundancy Problems

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>W</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>7</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

Update anomaly: What if we change W in this tuple only?
Deletion anomaly: What if we delete all employees with rating 5?
Insertion anomaly: What if we want to insert an employee and don’t know the hourly wage for his or her rating? (or we get it wrong?)

Decomposing a Relation

• Redundancy can be removed by “chopping” the relation into pieces.

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>H</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>40</td>
<td>R W</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>30</td>
<td>8 10</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
<td>5 7</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
<td>Wages</td>
</tr>
</tbody>
</table>

We’ll see how a type of integrity constraint, called functional dependencies, is used to drive this process
Taming Schema Redundancy

- Integrity constraints, in particular *functional dependencies*, can be used to identify schemas with problems and to *suggest refinements*

- Main refinement technique: *decomposition*
  - E.g., replacing ABCD (via projection) with:
    - AB and BCD, or
    - ACD and ABD

- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - What problems (if any) does the decomposition cause?

---

Functional Dependencies

- Let X and Y be *sets of attributes* in a relation R

- A *functional dependency* (FD) has the form $X \rightarrow Y$

  - Can read $\rightarrow$ as "determines"

  

- If two tuples in R agree on all the attributes in X, they must *also* agree on all attributes in Y

- (More formally): A functional dependency $X \rightarrow Y$ holds over relation schema R if, for every allowable instance $r$ of R:
  - $t1 \in r$, $t2 \in r$, $\pi_X(t1) = \pi_X(t2)$ implies $\pi_Y(t1) = \pi_Y(t2)$

---

Keys (Review)

- A set of fields is a *key* (aka candidate key) for a relation if:
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.

- A candidate key is *minimal*.
  - If AB is a candidate key, then neither A nor B is a key on its own.

- Part 2 above false? A superkey is not necessarily minimal (although it could be)
  - If AB is a candidate key, then ABC, ABD, and even AB are superkeys.

---

Functional Dependencies (cntd)

- Where do FDs come from?
  - Real-world integrity constraints and semantics

- Keys redefined as FDs:
  - A set of attributes $K$ is a *key* for a relation $R$ if:
    1. $K \rightarrow$ all (other) attributes of $R$
    2. No proper subset of $K$ satisfies the above condition
    - $K$ is *minimal* (thus a candidate key)
Exercise 3: Constructing FDs

• What functional dependencies do you think would make sense for this application?

<table>
<thead>
<tr>
<th>Bar_name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer_name</td>
<td>type</td>
</tr>
<tr>
<td>name</td>
<td>price</td>
</tr>
</tbody>
</table>

| Bars | Sells | Beers |

Example: Using Inference Rules

• Suppose relation R has three attributes A, B, C

• Given FDs:
  A → B
  B → C

• Using reflexivity
  A → A, AB → A, etc.

• Using transitivity
  A → C

• Using augmentation
  AC → BC, AB → AC, AB → BC
Attribute Closure

• If we just want to check if a given FD $X \rightarrow Y$ is in $F^+$, then:

1) Compute the attribute closure of $X$ (denoted $X^+$) with respect to $F$
   • $X^+ = \text{Set of all attributes } A \text{ such that } X \rightarrow A \text{ is in } F^+$
   • Initialize $X^+ := X$
   • Repeat until no change to $X^+$:
     if $U \rightarrow V$ in $F$ such that $U$ is in $X^+$, then add $V$ to $X^+$

2) Check if $Y$ is in $X^+$

Q. How can attribute closure be used to find the keys of a relation?

Exercise 4

• Contracts($cid,sid,jid,did,pid,qty,value$), and:
  C is the primary key: $C \rightarrow CSJDPQV$
  Project purchases each part using single contract: $JP \rightarrow C$
  Dept purchases at most 1 part from a supplier: $SD \rightarrow P$

• Show that SDJ is a superkey for Contracts
  • $JP \rightarrow C$, $C \rightarrow CSJDPQV$ imply $JP \rightarrow CSJDPQV$
    (by transitivity) (shows that JP is a superkey)
  • $SD \rightarrow P$ implies $SDJ \rightarrow JP$
    (by augmentation)
  • $SDJ \rightarrow JP$, $JP \rightarrow CSJDPQV$ imply $SDJ \rightarrow CSJDPQV$
    (by transitivity) thus SDJ is a superkey

The Issue with Non-Key FDs

• Why does the FD $rating \rightarrow hourly_wages$ yield redundancy issues?

• $Rating$ is a non-key field, so there could be duplicate pairs of particular ($rating$, $hourly_wages$) in this relation

• By separating ($rating$, $hourly_wages$) into its own relation, we resolve redundancy!
  — Can regain the original data via natural join

“Normal” Forms for a Schema

• Idea: decompose relation to remove redundancy
  — Decomposition guided by FDs

• Boyce-Codd Normal Form (BCNF)
  — Simple conditions under which anomalies cannot occur

• BCNF definition:
  Relation $R$ with FDs $F$ is in BCNF if, for all $X \rightarrow A$ in $F^+$
  — $A \in X$ (a trivial FD), or
  — $X$ is a superkey for $R$

• In other words: $R$ is in BCNF if the only non-trivial FDs over $R$ are key constraints
Decomposition of a Relation Schema

- If relation not in a desired normal form, can be decomposed into multiple relations that each are in that normal form.
- A decomposition of R consists of replacing R by two or more relations such that:
  - Each new relation contains a subset of R’s attributes, and
  - Every attribute of R appears as an attribute of at least one of the new relations.

Problems with Decompositions

- There are three potential problems to consider:
  1) May be impossible to reconstruct the original relation! (Lossiness)
  2) Checking functional dependencies may require joins.
  3) Some queries become more expensive due to joins.
     - e.g., How much does Smiley earn?

Lossiness (#1) cannot be allowed
#2 and #3 are design tradeoffs
Must consider these issues vs. redundancy

Lossy vs. Lossless Decomposition

- Example schema:
  Oversees (ProjectId, EmployeeId, DepartmentId)
- FDs:
  - E → P (an employee oversees only one project)
  - D → P (a dept works on only one project)
  - E → D (an employee only works with one dept for these projects)
- Example instance of Oversees:

<table>
<thead>
<tr>
<th>Project</th>
<th>Employee</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Alice</td>
<td>Physics</td>
</tr>
<tr>
<td>Comet</td>
<td>Bob</td>
<td>Astronomy</td>
</tr>
<tr>
<td>Genomics</td>
<td>Carl</td>
<td>Biology</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
<td>Biology</td>
</tr>
</tbody>
</table>

Lossy vs. Lossless Decomp (cntd)

- Redundancy with the FD D → P

<table>
<thead>
<tr>
<th>Project</th>
<th>Employee</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Alice</td>
<td>Physics</td>
</tr>
<tr>
<td>Comet</td>
<td>Bob</td>
<td>Astronomy</td>
</tr>
<tr>
<td>Genomics</td>
<td>Carl</td>
<td>Biology</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
<td>Biology</td>
</tr>
</tbody>
</table>

- Proposed decomposition:

<table>
<thead>
<tr>
<th>Project</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Physics</td>
</tr>
<tr>
<td>Comet</td>
<td>Astronomy</td>
</tr>
<tr>
<td>Genomics</td>
<td>Biology</td>
</tr>
</tbody>
</table>

Redundancy?
Lossy vs. Lossless Decomp (cntd)

- Redundancy with the FD $D \rightarrow P$

<table>
<thead>
<tr>
<th>Project</th>
<th>Employee</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comet</td>
<td>Alice</td>
<td>Physics</td>
</tr>
<tr>
<td>Comet</td>
<td>Bob</td>
<td>Astronomy</td>
</tr>
<tr>
<td>Genomics</td>
<td>Carl</td>
<td>Biology</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
<td>Biology</td>
</tr>
</tbody>
</table>

Lossy vs. Lossless Decomp (cntd)

- Decomposition attempt #2:

<table>
<thead>
<tr>
<th>Department</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics</td>
<td>Alice</td>
</tr>
<tr>
<td>Comet</td>
<td>Physics</td>
</tr>
<tr>
<td>Astronomy</td>
<td>Bob</td>
</tr>
<tr>
<td>Biology</td>
<td>Carl</td>
</tr>
<tr>
<td>Genomics</td>
<td>Denise</td>
</tr>
</tbody>
</table>

Lossy vs. Lossless Decomp (cntd)

- Decomposition of $R$ into $X$ and $Y$ is lossless-join w.r.t. a set of FDs $F$ if, for every instance $r$ that satisfies $F$:
  \[ \pi_X(r) \bowtie \pi_Y(r) = r \]

- Decomposition of $R$ into $X$ and $Y$ is lossless with respect to $F$ if and only if $F^+$ contains:
  \[ X \cap Y \rightarrow X, \text{ or } X \cap Y \rightarrow Y \]

  **Corollary**: If $Z \rightarrow W$ holds over $R$ and $Z \cap W$ is empty, then decomposition of $R$ into $R-W$ and $ZW$ is loss-less.

- In "Oversees" example, decomposing into \{E,P\} and \{D,P\} is lossy because the intersection (i.e., Project) is not a key of either resulting relation

Loss-less Decomposition

- Decomposition of $R$ into $X$ and $Y$ is lossless-join w.r.t. a set of FDs $F$ if, for every instance $r$ that satisfies $F$:
  \[ \pi_X(r) \bowtie \pi_Y(r) = r \]

- Decomposition of $R$ into $X$ and $Y$ is lossless with respect to $F$ if and only if $F^+$ contains:
  \[ X \cap Y \rightarrow X, \text{ or } X \cap Y \rightarrow Y \]

  **In other words, the common attributes form a key for $X$ or $Y$**

**Corollary**: If $Z \rightarrow W$ holds over $R$ and $Z \cap W$ is empty, then decomposition of $R$ into $R-W$ and $ZW$ is loss-less.

- In “Oversees” example, decomposing into \{E,P\} and \{D,P\} is lossy because the intersection (i.e., Project) is not a key of either resulting relation

Loss-less Decomposition into BCNF

- Relation $R$ has FDs $F$. If $Z \rightarrow W$ in $F$ violates BCNF:
  - decompose $R$ into $R-W$ and $ZW$
  (guaranteed to be loss-less)