CS 133: Databases

Fall 2019
Lec 21 – 11/21
Database Design
Prof. Beth Trushkowsky

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

Warm-up Exercise

(See exercise sheet. You can start before class.)

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))

Answer: Doesn’t enforce that each Bar has a tuple in Owns.
Table and Column constraints (SQLite)

Examples: Create Table

CREATE TABLE Friends (  
friend1 VARCHAR(40),  
friend2 VARCHAR(40),  
PRIMARY KEY(friend1,friend2),  
CONSTRAINT notSame CHECK (friend1 <> friend2)  );
CREATE TABLE Bar_Owns(  
name VARCHAR(40) PRIMARY KEY,  
address VARCHAR(40) NOT NULL,  
phone CHAR(12) DEFAULT "555-555-5555",  
owner CHAR(11) NOT NULL,  
FOREIGN KEY owner REFERENCES Drinkers(ssn)  
on DELETE NO ACTION  );

For cross-relation constraints, need assertion statement!

Triggers

- Trigger: procedure that starts automatically if specified changes occur to the DBMS

- Example, SQL:1999 syntax:

```
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
```
**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 as Bar_Sells?

**Example: Hourly_Emps**

- Consider a relation obtained from Hourly_Emps: `Hourly_Emps (ssn, name, lot, rating, wage_per_hr, hrs_per_wk)`
  - Assume we know, from application semantics:
    - `ssn` uniquely identifies an employee (is a key)
    - An employee’s `rating` determines their `wage_per_hr`

**Schema Refinement**

- Start with initial relational schema, either from scratch or from E/R modeling.
- Schema refinement objective: *could there be issues caused by data redundancy?*
  - Next: why redundancy is “bad”

**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>10</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>10</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>7</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>7</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>10</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

**Note on notation:** can denote a relation schema by listing its attributes, e.g., `SNLRWH` → the set of attributes `{S,N,L,R,W,H}`

**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 their 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>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
<td>5</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></td>
</tr>
</tbody>
</table>

Hourly_Emps2

We’ll see how a type of integrity constraint, called functional dependencies, is used to drive this “chopping” 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, or
    - Etc.

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

Keys (Review)

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

Q. Consider relation $R(\mathbf{a}, \mathbf{b}, \mathbf{c})$. For a fixed setting of a and b values, how many different c values could there be?

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

- Let $X$ and $Y$ be sets of attributes in a relation $R$
- A functional dependency (FD) has the form $X \rightarrow Y$
- If two tuples in $R$ have same values for all attributes in $X$, then they must also have same values for all attributes in $Y$

Can read “$\rightarrow$” as “determines.”

(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 $\pi_Y(t1) = \pi_Y(t2)$
Functional Dependencies (cntd)

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

• Keys redefined as FDs with set of attributes K and relation R:
  1. if \( K \rightarrow \text{all (other) attributes of } R \)
     \( K \) is a “super key”
  2. And if no proper subset of \( K \) satisfies the above condition, then
     \( K \) is minimal (and thus a candidate key)

Reasoning About FDs

• Given some FDs, can usually infer additional FDs that are true
  E.g., College use case:

<table>
<thead>
<tr>
<th>left side</th>
<th>right side</th>
</tr>
</thead>
<tbody>
<tr>
<td>profId</td>
<td>profName</td>
</tr>
<tr>
<td>profId</td>
<td>dept</td>
</tr>
</tbody>
</table>

implies:

<table>
<thead>
<tr>
<th>left side</th>
<th>right side</th>
</tr>
</thead>
<tbody>
<tr>
<td>profId, dept</td>
<td>profName, dept</td>
</tr>
</tbody>
</table>

Union rule (opposite: decomposition)

<table>
<thead>
<tr>
<th>left side</th>
<th>right side</th>
</tr>
</thead>
<tbody>
<tr>
<td>dept</td>
<td>building</td>
</tr>
</tbody>
</table>

implies:

<table>
<thead>
<tr>
<th>left side</th>
<th>right side</th>
</tr>
</thead>
<tbody>
<tr>
<td>profId, building</td>
<td>profName</td>
</tr>
</tbody>
</table>

Transitivity rule

• However, building, roomNum \( \rightarrow \) profName
  does NOT imply building \( \rightarrow \) profName or roomNum \( \rightarrow \) profName

• A particular FD \( f \) is implied by a set of FDs \( F \) if \( f \) holds
  whenever all FDs in \( F \) hold

Exercise 3: Constructing FDs

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

![Diagram with entities: Bars, Beers, Sells, name, address, price, type]

Bar_name \( \rightarrow \) address
beer_name \( \rightarrow \) type
bar_name, beer_name \( \rightarrow \) price

Closure and Rules of Inference

• \( F^* = \text{closure of } F \) is the set of all FDs that are implied by \( F \)
  (includes “trivial dependencies”: \( \text{RHS} \subseteq \text{LHS} \))

• Armstrong’s Axioms (\( X, Y, Z \) are sets of attributes):
  — 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 \( X \rightarrow Z \)

• Some additional rules (that 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: Using Inference Rules

• Suppose relation R has three attributes A,B,C and these FDs:
  \[ A \rightarrow B \]
  \[ B \rightarrow C \]

• Using reflexivity
  \[ A \rightarrow A, AB \rightarrow A, \text{etc.} \]

• Using transitivity
  \[ A \rightarrow C \]

• Using augmentation
  \[ AC \rightarrow BC, AB \rightarrow AC, AB \rightarrow BC \]

Attribute Closure

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

  1) Compute the attribute closure of \( X \) (denoted \( X^+ \)) with respect to \( F \)
     • \( X^+ = \) Set of all attributes \( A \) such that \( X \rightarrow A \) 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 \( X^+ \)

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

Q. How can attribute closure be used to determine if a set of attributes is a key for a relation?