Schema Refinement

Yanlei Diao
UMass Amherst

Example (Contd.)

<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>5</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>

- Rating (R) determines hourly wages (W):
  - **Redundant storage**
  - **Update:** Can we change W in just the 1st tuple of rating 8?
  - **Insertion:** Insert an employee without knowing the hourly wage for his rating? Insert the hourly wage for rating 10 with no employee?
  - **Deletion:** What if we have deleted all employees with rating 5.

The Evils of Redundancy

- Redundant storage causes several operation anomalies:
  - Insert and delete anomalies
  - **Functional dependencies**, a new type of integrity constraint, can be used to identify schemas with such problems.
  - IC’s we have seen: domain constraints, key constraints, foreign key constraints, general constraints (checks or assertions)
  - A new type of IC, functional dependencies, which can be checked using SQL checks or assertions.

The Evils of Redundancy

- Redundant storage causes several operation anomalies:
  - Insert and delete anomalies
  - **Functional dependencies**, a new type of integrity constraint, can be used to identify schemas with such problems.
  - IC’s we have seen: domain constraints, key constraints, foreign key constraints, general constraints (checks or assertions)
  - A new type of IC, functional dependencies, which can be checked using SQL checks or assertions.

Revisit a Previous Example

- Consider relation obtained from **Hourly_Emps**:
  - Hourly_Emps(see, name, lot, rating, hrly_wages, hrs_worked)
  - Denote the schema by listing all its attributes: SNLRWH

Will Two Smaller Tables be Better?

```plaintext
<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>5</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>
```

```plaintext
<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
<td>10</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>
```

1. Functional Dependencies (FDs)

- A **functional dependency** X \( \rightarrow \) Y holds over relation R if:
  - X and Y are two sets of attributes of R;
  - \( \forall \) allowable instance \( r \) of R:
  - \( \exists \) \( t_1, t_2 \in r, X \subset t_1, \) \( Y \subset t_2 \) implies \( X \subset t_2 \)
  - E.g., X = [R], Y = [W], and R \( \rightarrow \) W

- An FD holds for all allowable instances of a schema.

- Key constraint is a special form of FD:
  - K is a super key for R means that K \( \rightarrow \) R.
  - K \( \rightarrow \) R does not require K to be minimal!
FDs in the Hourly_Emps Example

- **Hourly_Emps**([sn, name, office, rating, hrly_wages, hrs_worked])
  - Denoted by SNLRWH
- Some FDs on Hourly_Emps:
  - `sn` is the key: \(S \rightarrow SNLRWH\)
  - `rating` determines `hrly_wages`: \(R \rightarrow W\)

Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:
  - \([sn \rightarrow did, did \rightarrow building]\) implies \(sn \rightarrow building\)
- Given a set of FDs \(F\), closure of \(F\) (\(F^+\)) is the set of all FDs that are implied by \(F\).
  - All FDs in \(F^+\) hold over the relation \(R\).

Axioms and Rules

- 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 \(X \rightarrow Z\)
- Couple of 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 \(X \rightarrow Z\)
- Computing the closure \(F^+\) using the axioms/rules:
  - Compute for all FD's.
  - Size of closure is exponential in number of attributes!

Attribute Closure

- What if we just want to check if a given FD \(X \rightarrow Y\) is in \(F^+\)?
- Simple algorithm for attribute closure \(X^+\):
  - \(X^+ := \{X\}\)
  - DO if there is \(U \rightarrow V\) in \(F\), s.t. \(U \subseteq X^+\), then \(X^+ = X^+ \cup V\)
  - UNTIL no change
- Does \(F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D \rightarrow E\}\) imply \(A \rightarrow E\)?
  - Is \(A \rightarrow E\) in the closure \(F^+\)?
  - Equivalently, is \(E\) in \(A^+\)?
Boyce-Codd Normal Form (contd.)

- Can we infer the value marked by ‘?’
  - If \( X \rightarrow A \), then the relation is not in BCNF
  - A reln. in BCNF can’t have \( X \rightarrow A \)
- Relation in BCNF:
  - Every field of every tuple records information that can’t be inferred using FD’s from other fields.
  - No redundancy can be detected using FDs!

Another Example

- When is redundancy possible?
  - Reserves( Sailor, Boat, Date, Credit_card) with
    S \( \rightarrow \) C, C \( \rightarrow \) S
  - Keys are C and CBD.
  - It is not in BCNF.
  - For each reservation of sailor S, same (S, C) is stored.

3. Decomposing a Relation Scheme

- A decomposition of R breaks R into two or more relns s.t.
  - Each new reln contains a subset of the attributes of R.
  - Every attribute of R appears in at least one new reln.
- Decompositions should be used only when:
  - R has redundancy related problems (not in BCNF),
  - We can afford the joins in queries later (performance penalty).

Example Decomposition

- Hourly_Emps (SNLRWH)
  - FDs: S \( \rightarrow \) SNLRWH and R \( \rightarrow \) W.
  - R \( \rightarrow \) W violates BCNF.
  - And it causes repeated (R,W) storage.
- To fix this, create a relation RW, remove W from the main schema. (SNLRWH) \( \rightarrow \) (SNLRH) and (RW).

(1) Lossless Join Decompositions

- Decomposition of schema R into R1 and R2 is **lossless-join** w.r.t. a set of FDs F if any reln. instance \( r \) that satisfies F:
  - \( r = \pi_{R1}(r) \bowtie \pi_{R2}(r) \)
- It is always true that \( r \subseteq \pi_{R1}(r) \bowtie \pi_{R2}(r) \).
  - A bad decomposition can cause \( r \subseteq \pi_{R1}(r) \bowtie \pi_{R2}(r) \).

A Simple Test for Lossless Join

- **Theorem**: Decomposition of R into R1 and R2 is **lossless-join** w.r.t. F if the \( F^+ \) contains:
  - \( R1 \cap R2 \rightarrow R1 \) or \( R1 \cap R2 \rightarrow R2 \)
  - (intersection of R1 and R2 is a (super) key of one of them.)
- Algorithm for a lossless join is based on the above result:
  - If \( U \rightarrow V \) holds over R and violates a BCNF definition, the decomposition into UV and R - V is lossless-join.
Dependency Preserving Decomposition

- **Contracts**: (Contractid, Supplierid, projectid, Deptid, Partid, Qty, Value), CSJDQV, with FDs:
  - C is key.
  - JP \( \rightarrow \) C: a project buys a given part using a single contract.
  - SD \( \rightarrow \) P: a department buys at most one part from a supplier.

What are the keys? Is it in BCNF?

- Keys: C, JP, SDJ.
- Not in BCNF due to SD \( \rightarrow \) P.

Lossless-join BCNF decomposition: SDP, CSJDQV

- Problem: Checking JP \( \rightarrow \) C requires an assertion (using join).
- How to write this assertion in SQL?

Algorithm for Decomposition into BCNF

- Relation R with FDs F. If X \( \rightarrow \) Y violates BCNF, decompose R into R1 = XY and R2 = R - Y.
  - For each Ri, compute F_{Ri} and check if it is in BCNF.
  - If not, pick a FD violating BCNF and keep composing Ri.

Repeated application of this process yields a lossless join decomposition into BCNF relations.
- But not necessarily dependency-preserving! So need to check whether some FDs are lost.

Steps of BCNF Decomposition

- Contracts(CSJDQV), key C, JP \( \rightarrow \) C, SD \( \rightarrow \) P, J \( \rightarrow \) S.
  2. Normal form. Not in BCNF; SD \( \rightarrow \) P and J \( \rightarrow \) S violate BCNF.
  3. Decomposition: To deal with SD \( \rightarrow \) P, decompose into SDP, CSJDQV.
     - SDP is in BCNF. But CSJDQV is not because:
     1. Projection of FDs and keys. Projection of FDs: keys C and DJ, J \( \rightarrow \) S.
     2. Normal form. Not BCNF; J \( \rightarrow \) S violates BCNF.
     3. Decomposition. For J \( \rightarrow \) S, decompose CSJDQV into JS and CJDQV.
     - JS is in BCNF. So is CJDQV.

- If several FDs violate BCNF, the order of “dealing with” them could lead to very different sets of relations!