Schema Refinement and Normal Forms

Yanlei Diao
UMass Amherst
Consider an Example

Consider relation obtained from \textbf{Hourly\_Emps}:
- \textbf{Hourly\_Emps}(\textit{ssn, name, lot, rating, hrly\_wages, hrs\_worked})
- Denote the schema by listing all its attributes: \textbf{SNLRWH}
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:** Delete all employees with rating 5.
### Will Two Smaller Tables be Better?

<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>

**Hourly_Emps2**

<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>

**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>
The Evils of Redundancy

- Redundant storage causes several operation anomalies:
  - Insert/delete/update anomalies

- *Functional dependencies*, a new type of integrity constraint, can be used to identify schemas with such problems.
  - IC’s we have seen: attribute constraints, key constraints, foreign key constraints, general constraints
  - A new type of IC: *functional dependencies*
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 $R$:
  $$t_1 \in r, t_2 \in r, \pi_X(t_1) = \pi_X(t_2) \text{ implies } \pi_Y(t_1) = \pi_Y(t_2)$$
- An FD holds for all allowable instances of a schema.
- Key constraint is a special form of FD:
  - $K$ is a candidate 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**(ssn, name, office, rating, hrly_wages, hrs_worked)
  - Denoted by SNLRWH

- Some FDs on Hourly_Emps:
  - *ssn* 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:
  - \( ssn \rightarrow did, \ did \rightarrow building \) implies \( ssn \rightarrow building \)

- 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 $X \rightarrow Z$

- A few 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 attrs!
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
- Check if a given FD $X \rightarrow Y$ is in $F^+$:
  - Simply check if $Y \subseteq X^+$.
- 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 $A^+$?
Normal Forms

- Role of FDs in detecting redundancy: R(A, B, C)
  - No FDs hold: No redundancy here.
  - Given A → B: Two tuples have the same A value will have the same B value!

- Normal forms: If a reln does not have certain kinds of FDs, certain redundancy-related problems are known not to occur.
Boyce-Codd Normal Form (BCNF)

- Rewrite every FD in the form of $X \rightarrow A$, $X$ is a set of attributes, $A$ is a **single** attribute
  - Use the decomposition rule

- Reln $R$ with FDs $F$ is in **BCNF** if $\forall X \rightarrow A$ in $F^+$:
  1. $A \in X$ (called a **trivial** FD), or
  2. $X$ is a **superkey** (i.e., contains a key) for $R$.

- In BCNF, the only non-trivial FDs are key constraints!
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 $X \rightarrow A$

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>$y_1$</td>
<td>a</td>
</tr>
<tr>
<td>x</td>
<td>$y_2$</td>
<td>?</td>
</tr>
</tbody>
</table>

- 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!*
Third Normal Form (3NF)

- Reln R with FDs F is in 3NF if \( \forall X \rightarrow A \) in \( F^+ \):
  1. \( A \in X \) (called a trivial FD), or
  2. \( X \) is a superkey for R, or
  3. \( A \) is part of some key for R (minimality of a key is crucial in the third condition).

- If R is in BCNF, obviously in 3NF.
Third Normal Form (contd.)

- If R is in 3NF, some *redundancy* is possible!
  - **Reserves** (Sailor, Boat, Date, Credit_card) with
    - S → C, C → S
  - Keys are SBD and CBD.
  - It is in 3NF.
  - But for each reservation of sailor S, same (S, C) is stored.
More on BCNF and 3NF

- 3NF is a weaker normal form than BCNF.
  - BCNF: no *redundancy* w.r.t. FDs. Not true for 3NF.
  - Possible to have *lossless-join, dependency-preserving* decomposition of R into 3NF. Not true for BCNF!

- To check if a reln R is in BCNF or 3NF, need to compute *all* the keys of R.

- To enforce FDs in BCNF or 3NF, declare key constraints and checks in CREATE TABLE.
  - *Reserves*(Sailor, Boat, Date, Credit_card) with
    
    \[ S \rightarrow C, \ C \rightarrow S \]
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.
Example Decomposition

- **Hourly_Emps (SNLRWH)**
  - FDs: $S \rightarrow SNLRWH$ and $R \rightarrow W$.
  - $R \rightarrow W$ violates 3NF.
  - And it causes repeated $(R,W)$ storage.

- To fix this, create a relation $RW$, remove $W$ from the main schema. $(SNLRWH) \rightarrow (SNLRH) \text{ and } (RW)$. 
Lossless Join Decompositions

- Decomposition of R into R1 and R2 is lossless-join w.r.t. a set of FDs F if ∀ instance r that satisfies F:
  - \( r = \pi_{R1}(r) \Join \pi_{R2}(r) \)
- It is always true that \( r \subseteq \pi_{R1}(r) \Join \pi_{R2}(r) \).
  - A bad decomposition can cause \( r \subsetneq \pi_{R1}(r) \Join \pi_{R2}(r) \).
A Simple Test for Lossless Join

- Decomposition of R into R1 and R2 is lossless-join wrt F iff the F⁺ contains:
  - R₁ ∩ R₂ → R₁ or R₁ ∩ R₂ → R₂
  - Intersection of R₁ and R₂ is a (super) key of one of them.

- How to apply this result?
  - If U → 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), CSJDQPV, with FDs:
  - C is key.
  - JP → C: a project buys a given part using a single contract.
  - SD → P: a department buys at most one part from a supplier.

- What are the keys? Which normal form is it in?
  - C, JP, SDJ. 3NF.

- Lossless-join BCNF decomposition: CSJDQV, SDP
  - Problem: Checking JP → C requires an assertion (using join)!
Dependency Preserving Decomposition

- The *projection* of a FD set \( F \) onto a decomposed reln \( R_1 \):
  - all \( U \rightarrow V \) s.t. (a) \( U, V \) are both in \( R_1 \), (b) \( U \rightarrow V \) is in closure \( F^+ \).
  - \( F_{R_1} = F^+_{R_1} \)

- Decomposition of \( R \) into \( R_1, R_2 \) is *dependency preserving* if
  \[
  (F_{R_1} \cup F_{R_2})^+ = F^+
  \]

- Important to consider \( F^+ \) *(not \( F \!\!\!\!\!\!\!\!\!\!\)*) in this definition:
  - ABC, \( A \rightarrow B, B \rightarrow C, C \rightarrow A \), decomposed into AB and BC.
  - Is this dependency preserving? Is \( C \rightarrow A \) preserved?
Decomposition into BCNF

- Relation R with FDs F. If $X \rightarrow Y$ violates BCNF, decompose R into $R_1 = R - Y$ and $R_2 = XY$.
  - For each $R_i$, compute $F_{R_i}$ and check if it is in BCNF.
  - If not, pick a FD violating BCNF and keep composing $R_i$.

- Repeated application of this process yields a lossless join decomposition into BCNF relations.
Steps of BCNF Decomposition

- Contracts(CSJDPQV), key C, JP → C, SD → P, J → S.
  1. **Keys and FDs.** Keys: C, JP, DJ. FDs: J → S.
  2. **Normal form.** Not in 3NF; SD → P and J → S violate BCNF.
  3. **Decomposition.** To deal with SD → 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 → S.
  2. **Normal form.** Not BCNF; J → S violates BCNF.
  3. **Decomposition.** For J → 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!
BCNF and Dependency Preservation

- Is a lossless-join BCNF decomposition dependency-preserving?
  - CSJDPQV with JP $\rightarrow$ C, SD $\rightarrow$ P and J $\rightarrow$ S.
  - CSJDPQV $\rightarrow$ SDP, JS and CJDQV
  - What about JP $\rightarrow$ C?
  - Adding JPC as a new relation to preserve JP $\rightarrow$ C introduces redundancy across relations and more joins
  - If we also have J$\rightarrow$C, JPC is not in BCNF.

- In general, there may not exist a lossless join, dependency-preserving decomposition into BCNF.
  - But there is always a lossless join, dependency-preserving decomposition into 3NF.